RENESAS Tool News

RENESAS TOOL NEWS on June 16, 2008: 080616/tn6

Notes on Using the Real-Time OSes --M3T-MR308/4 and M3T-MR30/4-- for the M16C MCU Family

Please take note of the following problems in using the real-time OSes-- M3T-MR308/4 and M3T-MR30/4--for the M16C MCU family:

- With using mailboxes with the TA_MPRI property (NOTE 1)
- With calculating stack usage

  NOTE:
  1. TA_MPRI is the mailbox property for arranging the messages in a queue in priority order.

---

1. Problem with using Mailboxes with the TA_MPRI Property

1.1 Products and Versions Concerned

(1) Real-time OS--M3T-MR308/4--for the M32C series (NOTE 2)
V.4.00 Release 00 through V.4.00 Release 02, and
V.4.00 Release 02A

(2) Real-time OS--M3T-MR30/4--for the M16C series (NOTE 3)
V.4.00 Release 00

NOTES:
2. The M32C series is the generic name of the M32C/80 and M16C/80 series.
3. The M16C series is the generic name of the M16C/60, /30, /20, /10, /Tiny, and R8C/Tiny series.

1.2 Description

If the snd_mbx or isnd_mbx service call is issued to a mailbox with the TA_MPRI property to send messages, they may be incorrectly sent, and indefinite values be written to indefinite areas.

1.3 Conditions

This problem may occur if the following conditions are all satisfied:

(1) The snd_mbx or isnd_mbx service call is issued to a mailbox with the TA_MPRI property.

(2) Two or more messages are stored in the mailbox in (1).

(3) During the execution of the service call in (1), an interrupt is generated, the interrupt handler issues the iprcv_mbx service call, and messages stored in the mailbox in (1) are delivered.

1.4 Workarounds

Before and after issuing the snd_mbx or isnd_mbx service call, disable and enable interrupt respectively as follows:

(1) In the case of a task having issued the snd_mbx service call
Do not use the interrupt enable bit, but disable interrupts by changing the processor interrupt priority level IPL to the kernel interrupt masking level (OS interrupt disabled level); then enable interrupts by restoring the level.

```
-----------------------------------------
void task(VP_INT exinf)
{
. . . . . . . . . . . . .
/* Interrupt disabled */
#pragma ASM
; Shown below is an example where kernel interrupt masking level
; (OS interrupt disabled level) is 7.
  LDIPL   #7
  NOP
  NOP
  NOP
#pragma ENDASM
  snd_mbx(ID_mbx,(T_MSG *)&msg);
  /* Interrupt enabled */
#pragma ASM
; IPL value is usually 0 during execution of tasks.
  LDIPL   #0
  NOP
  NOP
  NOP
#pragma ENDASM
. . . . . . . .
}
-----------------------------------------
```

(2) In the case of the interrupt handler having issued the isnd_mbx service call

Avoid the problem in either of the following ways:

(a) Perform the same procedure as Workaround (1)

(b) Clear and set the interrupt enable bit to disable interrupts and enable them.

```
------------------------------------------
void inthand(void)
{
. . . . . . . . . .
/* Workaround (a) */
/* Interrupt disabled */
#pragma ASM
; Shown below is an example where kernel interrupt masking level
; (OS interrupt disabled level) is 7.
 LDIPL   #7
 NOP
 NOP
 NOP
#pragma ENDASM
 isnd_mbx(ID_mbx,(T_MSG *)&msg);
 /* Interrupt enabled */
#pragma ASM
; IPL resumes value just before interrupt handler issuing isnd_mbx.
 LDIPL   #3
 NOP
 NOP
 NOP
#pragma ENDASM
. . . . . . . .
 /* Workaround (b) */
 /* Interrupt disabled */
#pragma ASM
 FCLR  I
#pragma ENDASM
 isnd_mbx(ID_mbx,(T_MSG *)&msg);
 /* Interrupt enabled */
#pragma ASM
 FSET  I
 NOP
#pragma ENDASM
. . . . . . . . .
}
-------------------------------------------
```

Notice:
Depending on your MCU, place and remove NOP instructions after the LDIPL, FSET, and FCLR instructions according to the timing at which the IPL is changed by the execution of the LDIPL instruction and at which the status of the I flag is reflected by the execution of the FSET and FCLR instructions.

1.5 Schedule of Fixing the Problem

This problem have already been fixed in M3T-MR30/4 V.4.00 Release 01, which will be published on July 4, 2008.
As for M3T-MR308/4, it will be fixed in the next release.

2. Problem with Calculating Stack Usage

2.1 Product and Version Concerned

Real-time OS--M3T-MR30/4--for the M16C series (See NOTE 3, in 1.1)
V.4.00 Release 00

2.2 Description

In the User's manual, the amounts of stack usage of service calls using C-language-interfaced routines have been expressed less than the actual ones.

2.3 Corrections

2.3.1 Corrections to User's Manual

Modify descriptions of the values of the amounts of stack usage of service calls using C-language in Section 10.2, "Necessary Stack Size" as follows:

Service Call's Name	Read	For
get_pri	5	2
wai_flg	5	2
pol_flg	5	2
twai_flg	7	4
tsnd_dtq	5	2
rcv_dtq	5	2
prcv_dtq	5	2

trcv_dtq	5	2
rcv_mbx	5	2
prcv_mbx	5	2
trcv_mbx	5	2
get_mpf	5	2
pget_mpf	5	2
tget_mpf	5	2
pget_mpl	5	2
get_tid	5	2
vtsnd_dtq	5	2
vrcv_dtq	7	4
vprcv_dtq	7	4
vtrcv_dtq	7	4
iget_pri	5	2
ipol_flg	5	2
iprcv_dtq	5	2
iprcv_mbx	5	2
ipget_mpf	5	2
iget_tid	5	2
viprcv_dtq	7	4

2.3.2 Adjustments of the Results Calculated Using the Stack Size Calculation Utility

When stack usage is calculated using the stack size calculation utility, add the adjustment value shown below for the service call involved to the calculated value. When two or more service calls are used, the maximum of their adjustment values shall be added.

(1) Only vprcv_dtq used:
Adjustment value 7

(2) Either pget_mpl or prcv_dtq used:
Adjustment value 5

(3) Any of the following used:
wai_flg, twai_flg, tsnd_dtq, rcv_dtq, trcv_dtq, rcv_mbx, trcv_mbx,

get_mpf, tget_mpf, vtsnd_dtq, vrcv_dtq, vtrcv_dtq, iget_pri, ipol_flg, iprcv_dtq, iprcv_mbx, ipget_mpf, iget_tid, and viprcv_dtq
Adjustment value 3

Example 1:
When the calculated result of the stack usage of a task is 52 bytes, and the task has issued vprcv_dtq or vrcv_dtq, an adjustment value of 7 is added to 52, resulting in the correct usage being 59 bytes.

Example 2:
When the calculated result of the stack usage of an interrupt handler is 36 bytes, and the handler has issued iget_tid or iget_pri, an adjustment value of 3 is added to 36, resulting in the correct usage being 39 bytes.

2.4 Schedule of Fixing the Problems

This problems have already been fixed in M3T-MR30/4 V.4.00 Release 01, which will be published on July 4, 2008.