GRADE **A**

MESC TECHNICAL NEWS No.M16C-14-9805

Precautions Regarding Writing to M16C/60, M16C/61, M16C/62 and M16C/63 Group MCUs Interrupt Control Registers

1. Related devices

M16C/60, M16C/61, M16C/62 and M16C/63 groups

With the M16C/60 series MCU, setting the interrupt priority level and clearing the interrupt request bit in the interrupt control registers should be done with interrupt disabled.
Executing these operations while interrupt is enabled may result in unintended CPU operations.

2. Symptom

Changing the Interrupt priority LeVeL select bit (ILVL) and clearing the Interrupt Request bit (IR) in the Interrupt Control Registers (ICRs) while the Interrupt enable FLAG (I-FLAG) is "1" may result in unintended operations, such as BRK and other interrupts being generated.

3. Considerations for writing new program

It is recommended that the interrupts must be disabled by clearing the I-FLAG, before setting ILVL or clearing the IR bit in the ICRs.
In order to avoid the influence of the CPU pipeline, a certain number of instructions (eg. NOP) should be inserted between writing to the ICRs and setting the I-FLAG.

The number of instructions (NOPs) required is shown in TABLE.

4. Conditions to be checked for program already written

Please confirm that at least one condition is met for both actions listed below. If any one of the conditions is met, the symptom will not occur.

(1) When changing ILVL

- I-FLAG is "0". (Interrupt disabled) (*Note)
- The processor interrupt priority level (IPL) in the flag register is "7".
- The ILVL changes from a lower level than IPL to a higher level.
- The ILVL before and after the change is lower than IPL.
- The ILVL before and after the change is higher than IPL.
- It is obvious that the corresponding interrupt will not occur while changing the ILVL.

(2) When clearing the IR

- I-FLAG is "0". (Interrupt disabled) (*Note)
- The IPL in the flag register is "7".
- The ILVL during the operation is "0".
- The ILVL is lower than IPL.
- It is obvious that the corresponding interrupt will not occur while clearing the IR.

Note: In order to avoid the influence of the CPU pipeline, a certain number of instructions (eg. NOP) should be inserted between writing to ICRs and setting the I-FLAG.

The number of instructions required is showed in the TABLE.

	When not using HOLD function	When using HOLD function
Example 1	Two NOP instructions required	Four NOP instructions required
Example 2	No NOP instruction required (because there is dummy read)	
Example 3	No NOP instruction required	

5. Program examples

The program examples are described as follow:

(1) For assembler

Example 1:

```
INT_SWITCH1:
  FCLR   I           ; Disable interrupts.
  AND.B  #00h, 0055h ; Clear TA0IC int. priority level and int. request bit.
  NOP                ; Four NOP instructions are required when using HOLD function.
  NOP
  FSET   I           ; Enable interrupts.
```

Example 2:

```
INT_SWITCH2:
  FCLR   I           ; Disable interrupts.
  AND.B  #00h, 0055h ; Clear TA0IC int. priority level and int. request bit.
  MOV.W  MEM, R0     ; Dummy read.
  FSET   I           ; Enable interrupts.
```

Example 3:

```
INT_SWITCH3:
  PUSHC  FLG         ; Push Flag register onto stack
  FCLR   I           ; Disable interrupts.
  AND.B  #00h, 0055h ; Clear TA0IC int. priority level and int. request bit.
  POPC   FLG         ; Enable interrupts.
```

The reason why two NOP instructions (four when using the HOLD function) or dummy read are inserted before FSET I in Examples 1 and 2 is to prevent the interrupt enable flag I from being set before the interrupt control register is rewritten due to effects of the instruction queue.

(2) For C language

```
#pragma  ASM
 INT_SWITCH:
  FCLR   I
#pragma  ENDASM
  TA0IC  & =00 ;     /* Clear TA0IC int. priority level and int. request bit. */
#pragma  ASM
  NOP                ; Four NOP instructions are required when using HOLD function.
  NOP
  FSET   I
#pragma  ENDASM
```