GRADE A

MESC TECHNICAL NEWS

No. M16C-60-0009

Notes on USB control registers of M30240 Group

1. Affected devices
M30240 Group

2. Symptom
There is a possibility that using read-modify-write instructions on the USB Control and Status Registers (CSR) might cause incorrect data to be written to these registers

2.1 Description
The USB CSRs (listed in Table 1) contain control and status bits that can be changed by both software and hardware. It is possible that while performing a read-modify-write instruction you might miss a bit that gets changed by hardware during the instruction execution. In other words, after the CPU performs the read portion of the instruction, a control or status bit might be changed by hardware, and when the write back portion of the instruction is executed, this change could be overwritten by the previous value that was read.
A list of read-modify-write instructions is shown in Table 2.

Table 1. USB Control and Status Registers

Symbol	Name	Address
EP0CS	USB Endpoint 0 Control and Status Register	0311_{16}
EP1ICS	USB Endpoint 1 IN Control and Status Register	0319_{16}
EP1OCS	USB Endpoint 1 OUT Control and Status Register	$031A_{16}$
EP2ICS	USB Endpoint 2 IN Control and Status Register	0321_{16}
EP2OCS	USB Endpoint 2 OUT Control and Status Register	0322_{16}
EP3ICS	USB Endpoint 3 IN Control and Status Register	0329_{16}
EP3OCS	USB Endpoint 3 OUT Control and Status Register	$032A_{16}$
EP4ICS	USB Endpoint 4 IN Control and Status Register	0331_{16}
EP4OCS	USB Endpoint 4 OUT Control and Status Register	0332_{16}

Table 2. Read-modify-write Instruction

Function	Mnemonic
Bit Manipulation	BCLR, BNOT, BSET, BTSTC, BTSTS
Shift	ROLC, RORC, ROT, SHA, SHL
Arithmetic	ABS, ADC, ADCF, ADD, DEC, EXTS, INC, MUL, MULU, NEG, SBB, SUB
Logical	AND, NOT, OR, XOR
Jump	ADJNZ, SBJNZ

2.2 Example (See Figure 1)

IN_PKT_RDY bit in EP1ICS register is set to "1" by software indicating to the USB Function Control Unit (FCU) that a packet of data is ready to be sent out to the host. In this condition, the following is the explanation to set ISO bit to "1" using the BSET instruction.

(1) When the EP1ICS register is read for the BSET instruction, IN_PKT_RDY bit could still be "1".
(2) Now while this instruction is being executed, the packet of data is done being sent to the host, and the USB FCU (hardware) clears the IN_PKT_RDY bit to "0" indicating the transfer is complete.
(3) When the modified EP1ICS register with the ISO bit set to "1" from the BSET instruction is written back, the value written back to IN_PKT_RDY bit would be "1", because that's what was originally read. This will overwrite the "0" that was set by the hardware.
(4) Writing "1" back to IN_PKT_RDY bit will cause the USB FCU to send out another packet of data. In this example, the device would accidentally send out an extra USB packet to the host.

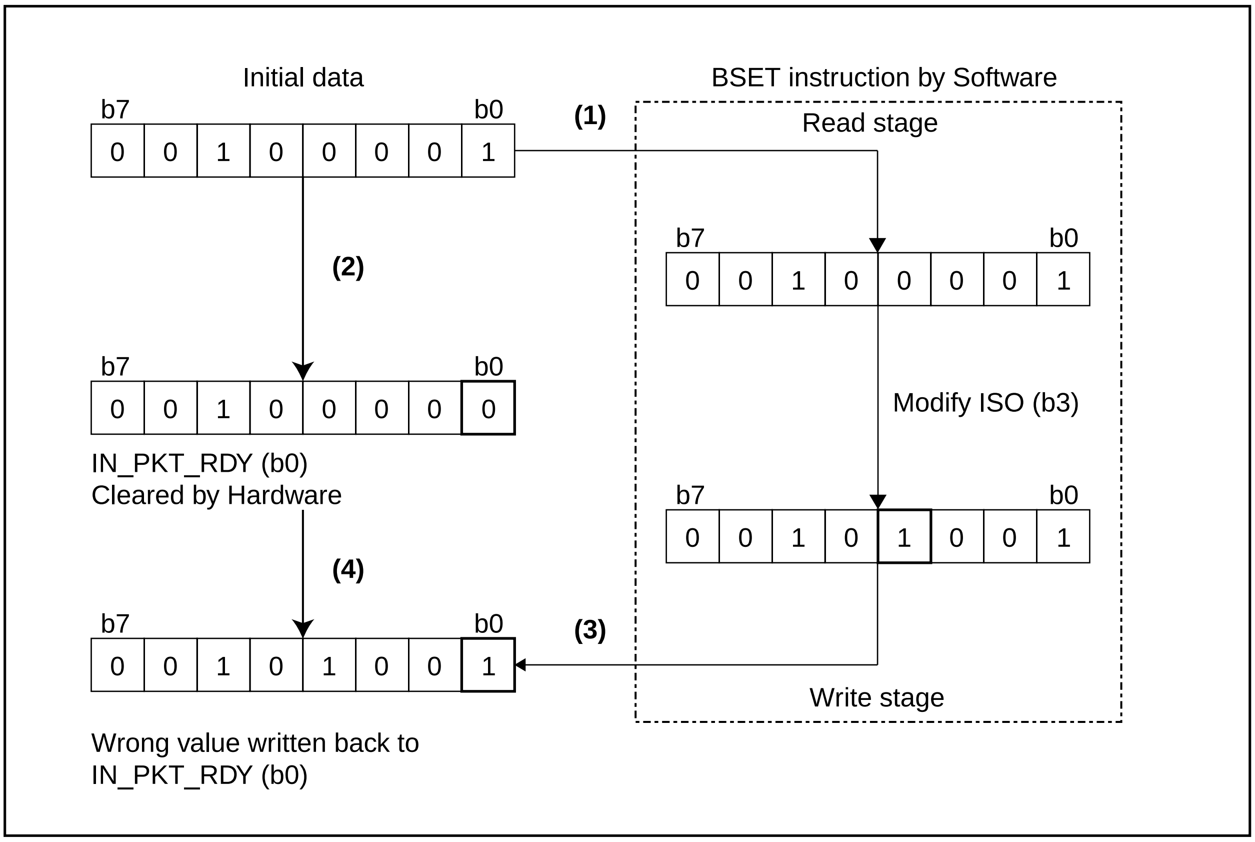

Figure 1. Example Error Condition

3. Countermeasure

Do not use read-modify-write instructions on these registers (see Table 2).
To change any bits in these registers, take the following procedure.
(1) First read the value of the register and save it to a variable or a data register.
(2) Change the target bit on the variable or the data register. At the same time, mask out any bits that you don't want to change by setting them to a value that won't change their state (see Table 3). For example: writing "0" to IN_PKT_RDY bit has no effect on it's state; writing "1" to the UNDER_RUN bit has no effect on it's state, etc.
(3) Then use a transfer instruction such as MOV instruction to write back to the register.
See Table 3 in the next page to identify which bits can be changed by both hardware and software and can potentially have incorrect data written to them in all of the USB control status registers.

The example shown in the next page can be used to avoid the possibility of writing incorrect bits for the situation described in Section 2. In this case we should write "0" to IN_PKT_RDY bit and "1" to UNDER_RUN bit to be sure and not change their states. These are the only two bits in the USB IN control status registers that can be changed by both hardware and software.

Table 3. Bits that can have incorrect data written back to them

Register Name	Bit Name (number)	Value to write for "No Change"
EP0CS	IN_PKT_RDY (b1)	"0"
	DATA_END (b3)	"0"
	FORCE_STALL (b4)	"1"
EPxICS (x = 1 to 4)	IN_PKT_RDY (b0)	"0"
	UNDER_RUN (b1)	"1"
EPxOCS (x = 1 to 4)	OUT_PKT_RDY (b0)	"1"
	OVER_RUN (b1)	"1"
	FORCE_STALL (b4)	"1"
	DATA_ERR (b5)	"1"

● **Example using C programming**

```
#pragma      ADDRESS          EP1ICS    0319h
char near Variable;

void  Func (void)
{
        :
        :
Variable = EP1ICS;             /* save EP1ICS register to temp variable */
Variable | = 0x0A;             /* set ISO bit and write "1" to UNDER_RUN bit (no change) */
Variable & = 0xFE;             /* write "0" to IN_PKT_RDY bit (no change)*/
EP1ICS = Variable;             /* write back to EP1ICS register */
        :
        :
}
```

● **Image of extract**

```
        :
        :
mov.b   EP1ICS,Variable        ; save EP1ICS register to temp variable
or.b    #0Ah,Variable          ; set ISO bit and write "1" to UNDER_RUN bit (no change)
and.b   #0FEh,Variable         ; write "0" to IN_PKT_RDY bit (no change)
mov.b   Variable,EP1ICS        ; write back to EP1ICS register
        :
        :
```

Figure 2. Example using C programming