AVR072: Accessing 16-bit I/O Registers

Features
- **Routines for Accessing 16-bits I/O Registers**
- **Macros Included for AVR- and IAR Assembler**

Description

All mid- and high-range AVR devices include a 16-bit Timer and some also include a 10-bit A/D Converter. These I/O modules have 16-bit registers that can be accessed from the AVR core using the *in* and *out* instructions. Since the AVR microcontroller has an 8-bit I/O bus the access to the I/O Register must be performed in two instruction cycles. An interrupt can occur between the instructions. If the interrupt function accesses the same resources (16-bit Timer or ADC) the 16-bit I/O Register access must be made an atomic operation, i.e., an operation that cannot be interrupted.

I/O modules with 16-bit Registers include a temporary register for the high byte (bit 15 to 8). Note that the 16-bit Timer (Timer1) has only one temporary register that is shared between all it's 16-bit register pairs. A 16-bit I/O read is normally done like this:

```
Cycle 1: in  r16, TCNT1L  ;Reading low byte into r16,this triggers the high
                          ; byte to be latched in the temporary shadow
                          ; register.
Cycle 2: in  r17, TCNT1H  ;Reading high byte from the temporary register.
```

Figure 1. 16-bit I/O Register Read[(1)]
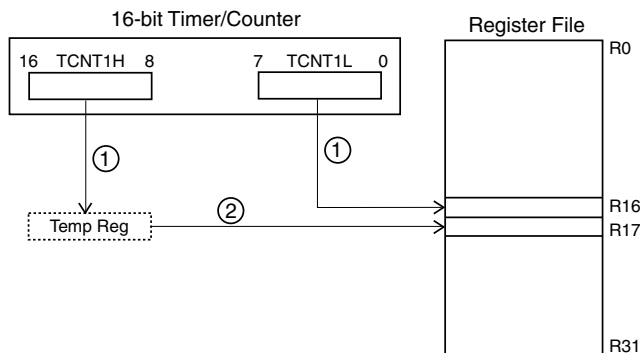
Note: 1. Circled numbers indicate cycle number.

If an interrupt occurs between the two instructions for any of the operations, and the interrupt handler access any of the Timer1's low bytes, the temporary register might change it's value. Returning from the interrupt the AVR core now reads the corrupted value into register r17.

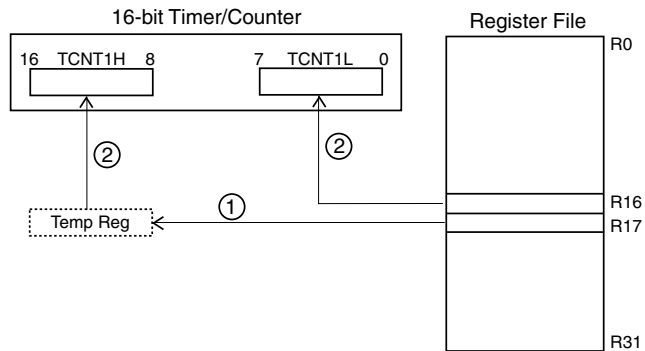A 16-bit I/O write is done as follow:

```
                              ;r17 contains the high byte while r16 contains
                              ;the low byte that is to be written.
Cycle 1: out   TCNT1H, r17   ;Writing the high byte to the temporary register.
Cycle 2: out   TCNT1L, r16   ;Writing both the low byte and the temporary
                              ;register into the I/O Register.
```

Figure 2. 16-bit I/O Register Write[1]

Note: 1. Circled numbers indicate cycle number.

Note that the read and the write operation differ in the order the high and low I/O Register is accessed. If the order is reversed, the high value will be incorrectly read or written.

Solution

To avoid the situations described above, the following macros for the AVR assembler and the IAR C compiler can be used

AVR Assembler Macros

```
.macro outw
  cli
  out    @0, @1
  out    @0-1, @2
  sei
.endmacro

.macro   inw
  cli
  in      @1, @2-1
  in      @0, @2
  sei
.endmacro
```

Usage

```
.include "8515def.inc"
inw     r17, r16, TCNT1H      ; Reads the counter value (high, low, adr)
outw    TCNT1H, r17, r16      ; Writes the counter value (adr, high, low)
IAR C MACROS
#include <ina90.h>

#define outw( ADDRESS, VAL )\
{\
    _CLI();
    ADDRESS = VAL;\
    _SEI();\
}

#define inw( ADDRESS, VAL )\
{\
    _CLI();
    VAL = ADDRESS;\
    _SEI();\
}
```

Usage

```
#include <io8515.h>

    inw( TCNT1, i )              ;/* Reads the counter value */
    outw( TCNT1, i )             ;/* Writes the counter value */
```

Notes

Notes: 1. The outw and the inw macros uses four instruction cycles which is the same amount of cycles as the ret instruction uses. Doing so the macros will not increase the worst case interrupt response time.
2. The macros will not work for setting the EEPROM address since the address can not be changed during write operation. This will lead to corruption of the data written.

Printed on recycled paper.