EE2016 Microprocessors Theory and Lab, Aug - Nov 2024.

Solutions to Tutorial 6 (Classes on last week of October, 2024)

Dr. R. Manivasakan, EE Dept, IIT Madras.

Portions: Timers/ Counter peripheral in AVR processor, configuration of timers and counters in AVR processors: Registers in timer/counter and their role in assembly AVR programming

1 Fill up the blanks:

1. Two major application modes in which the Timer / Counter hardware (TCH) in AVR microcontrollers could be used, are (a) timer and (b) counter
2. For operation of timer/counter hardware (TCH), as timer one needs internal crystal oscillator (internal crystal oscillator, external pulse source).
3. For operation of timer/counter hardware (TCH), as counter one needs external pulse source (internal crystal oscillator, external pulse source).
4. AVR ATmega32 has 3 number of timer/counter hardware (TCH) units.
5. Contents of the TCH could be accessed through TCNTn register.
6. When the counter associated with the AVR TCH rolls over, TOVn flag (in the TIFR register) is set.
7. AVR TCH registers are accessed through IN and OUT instructions.
8. Which bits in TCCR0 are used to select(s) the clock or prescaled clock or ground (disabling TCH) or rising / falling edge?

   MSB three bits in TCCR0, namely CS02, CS01 and CS00
9. What is CTC mode of an AVR TCH? Explain. CTC is the abbrevation for "Clear Timer on Compare Match" and the register OCRn (output compare register, n = 0,1,2,...) and flag OCFn (output compare flag) mechanisms to implement the CTC mode of operation.

2 Solve ALL the problems

1. **Timer:** An internally fed crystal oscillator (operating at 8 MHz) to AVR TCH counts from 0x3E till 0xFF. Compute the delay generated.

   Solutions: 0xFF corresponds to 255 and 0x3E corresponds to 62 and hence 0xFF - 0x3E = 255-62 = 193. Add 1 to it, to accommodate for roll over, which makes it 194 clock cycles. A single clock cycle is 0.125 $\mu Secs$. Hence the delay generated is $194 \times 0.125 = 24.25$ $\mu Secs$.
2. **50% Duty Cycle Square Wave Generation using AVR Timer/ Counter:** Write an AVR assembly program to generate a square wave of 50% duty cycle on the PORTB.5 bit. Timer0 is used to generate the time delay. Steps and constraints are
   (a) 0xF2 is loaded into TCNT0
   (b) TCCR0 is loaded with 0xF2 and Timer0 is started.

(c) Timer0 counts up with passing of each clock, which is provided by the crystal oscillator. At the time, it starts from F2, F3, ... 0xFF and one more clock rolls it to 0, raising the Timer0 flag (TOV0=1).

(d) Timer0 is stopped.

(e) TOV0 flag cleared and starts over again. Assume that XTAL = 8 MHz. Use "M32DEF.INC".

Calculate one period of the square wave. To know the number of cycles each instructions (used in above code) refer to Mazidi in page 695. Verify that the duty cycle is 50%. Calculate the frequency.

Solution:

```
                .INCLUDE "M32DEF.INC"
0       .MACRO INITSTACK                ;set up stack
1               LDI     R20, HIGH(RAMEND)       ; initialise stack
2               OUT     SPH, R20
3               LDI R20, LOW(RAMEND)
4               OUT SPL, R20                    ; initialization over
5       .ENDMACRO
6               INITSTACK
7               LDI R16, 1<<5           ;R16=0x20 (0010 0000 for PB5) stack
8               SBI DDRB, 5                     ;PB5 as an output
9               LDI R17, 0
10               OUT PORTB, R17                 ;clear PORTB
11 BEGIN:        RCALL DELAY                    ;call timer delay
12               EOR R17,R16                    ;toggle D5 of R17 by Ex-Oring with 1
13               OUT PORTB, R17                 ;toggle PB5
14               RJMP BEGIN
15 ;------ Time0 delay ------------
16 DELAY:        LDI R20, 0xF2                  ;R20 = 0xF2
17               OUT TCNT0, R20                 ;load timer0
18               LDI R20, 0x01
19               OUT TCCR0, R20                 ;Timer0, Normal mode, internal clk, no prescaler
20 AGAIN:        IN R20, TIFR                   ;read TIFR at each clk pulse - TCNT0 is incrm-
ntng
21               SBRS R20, TOV0                 ;if TOV0 is set skip next instruction
22               RJMP AGAIN
23               LDI R20,0x00
24               OUT TCCR0, R20                 ;stop Timer0
25               LDI R20, (1<<TOV0)
26               OUT TIFR, R20                  ;clear TOV0 flag by writing a 1 to TIFR
27               RET
```

In order to calculate one period of the cycle, one needs to calculate, the time T_{hlf} for one single run from 11 to 14. The required time is twice this. A single time run T_{hlf} consists of (a) the time for the counter to reach 0xFF from 0xF2 (b) one single run from 16 to 20, 22 to 27 and 12 to 14.

```
                                          Cycles
        LDI    R16,0x20
        SBI    DDRB,5
        LDI    R17,0
        OUT    PORTB,R17
BEGIN:  RCALL  DELAY                        3
        EOR    R17,R16                      1
        OUT    PORTB,R17                    1
        RJMP   BEGIN                        2
DELAY:  LDI    R20,0xF2                     1
        OUT    TCNT0,R20                    1
        LDI    R20,0x01                     1
        OUT    TCCR0,R20                    1
AGAIN:  IN     R20,TIFR                     1
        SBRS   R20,0                    1 / 2
        RJMP   AGAIN                        2
        LDI    R20,0x0                      1
        OUT    TCCR0,R20                    1
        LDI    R20,0x01                     1
        OUT    TIFR,R20                     1
        RET                                 4
                                           24
```

The point is that the moment that the line 19 (TCCR0 is assigned) counter running freely. Parallely, the "Again" loop is also running. This means that when the counter rolls over (14th cycle), the again loop has finished 3 loops (1+1+2) - 12 cycles. In the 4th loop, TOV0 is set to 1 and hence, the SBRS just misses the TOV0 and hence we have 14 (counter) + 4 (RJMP+LDI+SBRS) falls through. Making it a total of 18 cycles for 20 - 22 lines of the program. This is consistent with the results given in Mazidi (line 319). Hence the T = 9.5 musec and frequency is 105.263 kHz.

3. In mazidi, go through and understand AVR assembly program Ex 9.3 in p318. The same program, a part (after INITSTACK) has been rewritten in p319 (with explict number of clock cycles, to calculate the number of clock cycles in a half-wave period). Identify some lines were modified in p.319 Are those lines are correct? If they are, then justify.

   Solution: Not provided.

4. **Non 50% Duty Cycle Square Wave Generation using AVR Timer/ Counter:** In the previous problem the duty cycle is 50%. It is easy to generate, as it reuses the same subroutine for both ON and OFF periods. Come up with a strategy to generate a square wave with non 50% duty cycle using AVR Timer / Counter. Write the AVR program and calculate the duty cycle (You may assume the parameters as in the previous problem).

   Solution: In the most general form, have the following structure: RCALL 1stHlfPeriod. Toggle then RCALL 2ndHlfPeriod. Toggle. Within each half period define your own delay to meet the specifications including the duty cycle. Example program is given below

```
                .INCLUDE "M32DEF.INC"
0               .MACRO INITSTACK                ;set up stack
```

```
1                   LDI         R20, HIGH(RAMEND)      ; initialise stack
2                   OUT       SPH, R20
3                   LDI R20, LOW(RAMEND)
4                   OUT SPL, R20                          ; initialization over
5           .ENDMACRO
6                   INITSTACK
7                   LDI R16, 1<<5           ;R16=0x20 (0010 0000 for PB5) stack
8                   SBI DDRB, 5                 ;PB5 as an output
9                   LDI R17, 0
10                   OUT PORTB, R17              ;clear PORTB
11 BEGIN:            RCALL 1stHlfPeriod            ;call timer delay
12                   EOR R17, R16                ;toggle D5 of R17 by Ex-Oring with 1
13                   OUT PORTB, R17              ;toggle PB5
14                   RCALL 2ndHlfPeriod          ;call timer delay
15                   EOR R17, R16                ;toggle D5 of R17 by Ex-Oring with 1
16                   OUT PORTB, R17              ;toggle PB5
17                   RJMP BEGIN
18
15 ;------ 1stHlfPeriod time interval generation ------------
16 1stHlfPeriod:          LDI R20, 0x00             ;1stHlfPeriod is 256 clk cycles
17                   OUT TCNT0, R20              ;load timer0 0x00
18                   LDI R20, 0x01
19                   OUT TCCR0, R20              ;Timer0, Normal mode, internal clk, no prescaler
20 AGAIN:            IN R20, TIFR                ;read TIFR at each clk pulse - TCNT0 is incrm-
ntng
21                   SBRS R20, TOV0              ;if TOV0 is set skip next instruction
22                   RJMP AGAIN
23                   LDI R20,0x00
24                   OUT TCCR0, R20              ;stop Timer0
25                   LDI R20, (1<<TOV0)
26                   OUT TIFR, R20                 ;clear TOV0 flag by writing a 1 to TIFR
27                   RET
;------ 2ndHlfPeriod time interval generation ------------
16 2ndHlfPeriod:      LDI R20, 0x80             ;2ndHlfPeriod is 128 clk cycles
17                   OUT TCNT0, R20                ;duty cycle is 66.7%
18                   LDI R20, 0x01             ; One can have any arbitrary duty cycle this way
19                   OUT TCCR0, R20                ;Timer0, Normal mode, internal clk, no prescaler
20 AGAIN:            IN R20, TIFR                ;read TIFR at each clk pulse - TCNT0 is incrm-
ntng
21                   SBRS R20, TOV0              ;if TOV0 is set skip next instruction
22                   RJMP AGAIN
23                   LDI R20,0x00
```

```
24          OUT TCCR0, R20          ;stop Timer0
25          LDI R20, (1<<TOV0)
26          OUT TIFR, R20           ;clear TOV0 flag by writing a 1 to TIFR
27          RET
```

5. **AVR Timer/ Counter in CTC Mode:** Assuming Xtal frequency 8 MHz, write a program to generate a delay of 12.8 ms. Use Timer0 CTC mode, with prescaler = 1024. Use polling. Can you rewrite the AVR program which uses interrupt? How many cycles do, both take?

   Solution: Due to prescaler value of 1024, each period clock input is 1024 x 0.125 = 128 musec. Thus, in order generate a delay of 12.8 ms, we should wait for 12.8 ms/ 128 musec = 100 clocks. Therefore, the OCR0 register should be loaded with 100 -1 = 99

```
            .INCLUDE "M32DEF.INC"
            LDI R16, 0x08       ;
            SBI DDRB, 3         ;PB3 as an output
            LDI R17, 0          ;
            OUT PORTB, R17
DELAY:      LDI R20, 0x00
            OUT TCNT0, R20          ; start at 0
            LDI R20, 99
            OUT OCR0, R20           ; compare value 99
            LDI R20, 0x0D
            OUT TCCR0, R20      ; timer0, CTC mode, prescaler = 1024
AGAIN:      IN      R20, TIFR   ; read TIFR
            SRBS R20, OCF0 ; if compare value reached come out of lp
            RJMP AGAIN
            LDI R20, 0x00
            OUT TCCR0, R20
            LDI R20, 1<<OCF0
            OUT TIFR, R20           ;clear OCF0 flag
```

   Interrupt part is ignored. (And is taken up in the next problem).

6. **AVR Timer/ Counter with Interrupt:** Consider the AVR timer program Ex10.1 in page 370, which uses the interrupt feature of AVR. Related to this, one needs to go through the contents of the register TIMSK (Timer Interrupt Mask) Register given in p 368. Of particular interest is the LSB of this 8-bit register, namely, TOIE0 (Timer0 Overflow Interrupt Enable). Rest of the registers used in Ex 10.1 are the usual timer registers, TCNT0 and TCCR0. Important aspect is that this program uses interrupt, instead of polling (using SBRS instruction as in Ex 9.3).

   Problem here is to rewrite Ex 10.1 so that it uses polling, rather than interrupt.

   Solution: Due Hint: Include in the infinite loop in the main program the half wave generation along with toggling and required initializations. Use SBRS and RJMP for checking TOV0 bit. <I would upload the code, later. But you should try independently. Take this as an exercise>