Interrupts and Timers in AVR Atmega8

Jnaneswara Rao Rompilli
EE20B052

Aim:

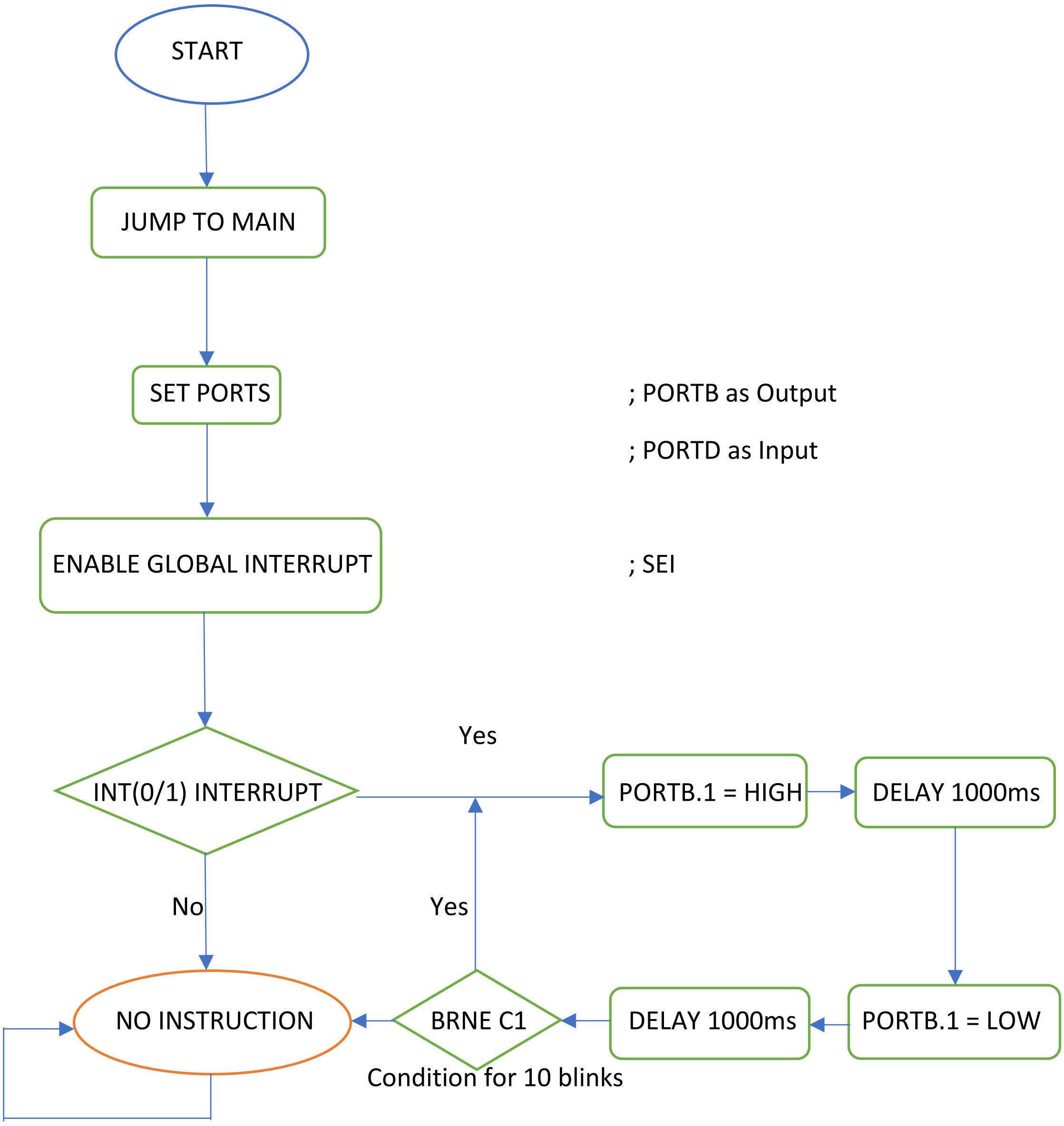
I. Generate an external hardware interrupt using an emulation of a push button switch.
II. Write an Instruction Service Routine (ISR) to blink an LED 10 times with duty cycle 50%.

Tasks:

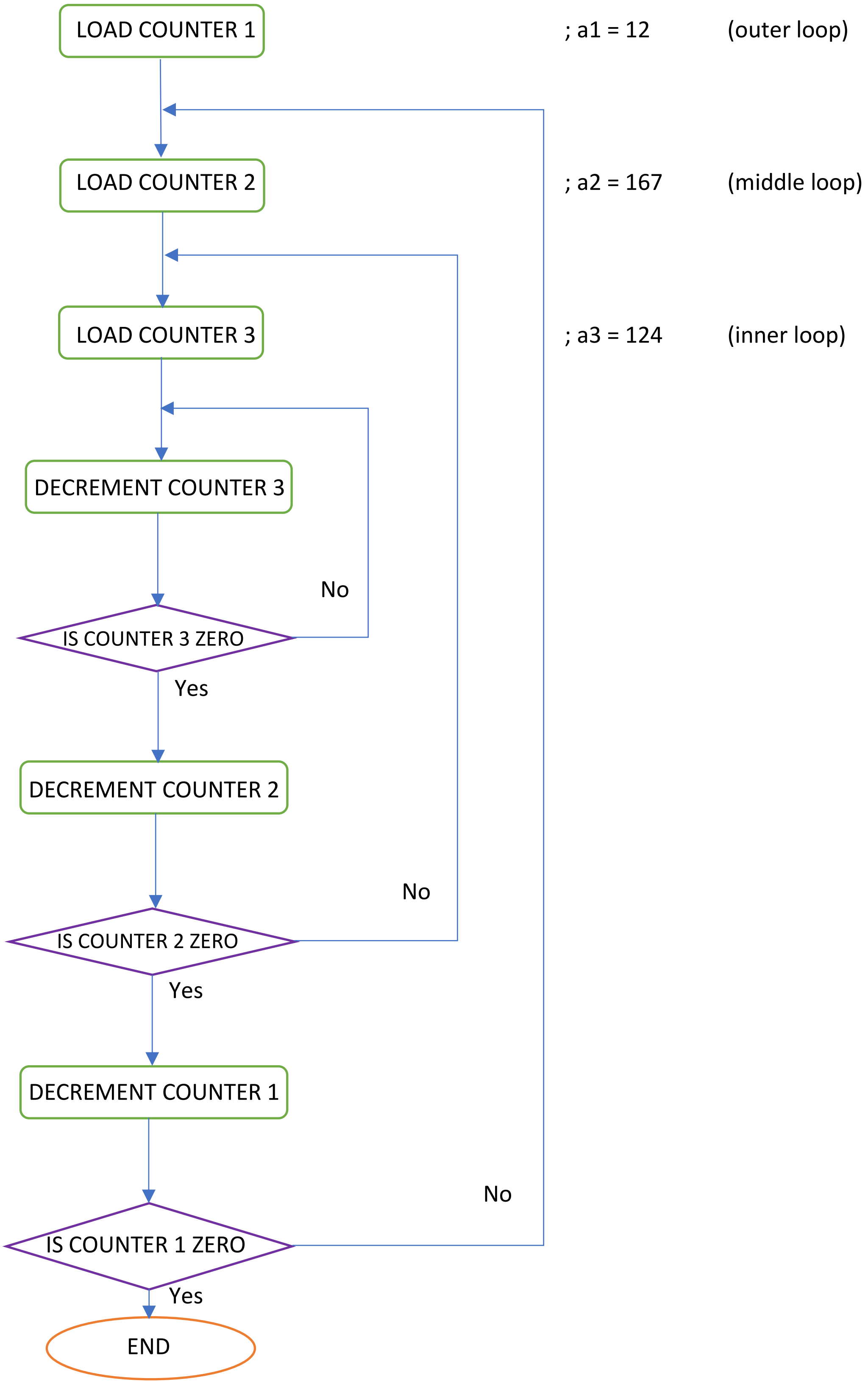
I. Use INT0 and INT1 blink LED 10 times with duty cycle 50% (ON/OFF – 1 second) on pushing the button.

Logic:

a) Flowchart for whole program:



b) Flow chart for 1 sec delay using loops:



After considering clock cycles required for each nested loop considering every instruction

- Clock Speed = 1MHz
- Total clock cycles = $((4 \times a3 + 3) \times a2) + 3) \times a1$ = n
- Modified a1, a2, a3 such n is as close as possible to 10^6 (cycles)
- a1 = 12, a2 = 167, a3 = 127
- On substituting a1, a2, a3 => n = 1,000,032 ≈ 1 sec, means 0.0032% error

Code:

1.1) Assembly program implementing LED blink from **INT0** hardware interrupt

```
.org 0x0000
rjmp reset

.org 0x0001
rjmp int0_ISR

.org 0x0100
reset:

              LDI R16, 0x70        ;Loading stack pointer address
              OUT SPL, R16
              LDI R16, 0x00
              OUT SPH, R16


              LDI R16,0x01         ; Interface port B pin0 to be output
              OUT DDRB, R16        ; so to view LED blinking

              LDI R16,0x00
              OUT DDRD, R16
              OUT PORTD, R16       ; PORTD.2 as Push Button - Input


              LDI R16,0x00         ; Set MCUCR register to enable low level interrupt
              OUT MCUCR, R16


              LDI R16,0x40         ; Set D6 Bit of GICR register to enable interrupt INT0
              OUT GICR, R16

              LDI R16,0x00         ; PORTB as Output
              OUT PORTB, R16

              SEI                  ;

ind_loop:     rjmp ind_loop

int0_ISR:     IN R16,SREG
              PUSH R16

              LDI R16,0x0A
              MOV R0,R16

c1:           LDI R16,0x01         ; To blink LED 10 times (R0 used)
              OUT PORTB,R16        ; Making LED - HIGH

              LDI R16,0x0C         ; Outer loop - 12 times
a1:           LDI R17,0xA7         ; Middle loop - 167 times
a2:           LDI R18, 0x7C        ; Inner loop - 124 times
a3:           DEC R18              ; Delay = 1000032 cycles - 0.0032% error
              NOP                  ; Time delay for this loop = 4*a1*a2*a3 + 3*a1*a2 + 3
              BRNE a3
              DEC R17
              BRNE a2
              DEC R16
              BRNE a1

              LDI R16,0x00
              OUT PORTB,R16        ; Making LED - HIGH
              LDI R16, 0x0C        ; 12 times
b1:           LDI R17,0xA7         ; 167 times
b2:           LDI R18, 0x7C        ; 124 times
b3:           DEC R18
              NOP
              BRNE b3
              DEC R17
              BRNE b2
              DEC R16
              BRNE b1

              DEC R0
              BRNE c1


              POP R16              ; Popping context from Stack
              OUT SREG,R16


              RETI
```

1.2) C program implementing LED blink from **INT0** hardware interrupt

```
  #define F_CPU 1000000 // clock frequency

  #include <avr/io.h>
  #include <util/delay.h>
  #include <avr/interrupt.h>

  ISR(INT0_vect)
  {
       int i;
       for (i = 1; i <= 10; i++) // for 10 times LED blink

       {
              PORTB = 0x01;
              // delay of 1 sec = 5*200ms
              for(int j = 1; j<=5; j++)
              {
                     _delay_ms(200) ;
              }
              PORTB = 0x00;
              // delay of 1 sec = 5*200ms
              for(int j = 1; j<=5; j++)
              {
                     _delay_ms(200) ;
              }

       }
}
int main(void)
{
       DDRD = 0x00;         // PORTD as input
       DDRB = 0x01;         // Make PB0 as output
       MCUCR = 0x00;        // Set MCUCR to level triggered
       GICR = 0x40;         // Enable interrupt INT0
       PORTB = 0x00;
       sei();               // global interrupt flag

       while (1)            //wait
       {
       }
}
```

Drive links for above programs:

- INT0 ASM
- INT0 C

2.1) Assembly program implementing LED blink from **INT1** hardware interrupt

```
.org 0x0000
rjmp reset

.org 0x0002
rjmp int1_ISR

.org 0x0100
reset:

              LDI R16,0x70         ;Loading stack pointer address
              OUT SPL, R16
              LDI R16,0x00
              OUT SPH, R16


              LDI R16,0x01         ; Interface port B pin0 to be output
              OUT DDRB, R16        ; so to view LED blinking

              LDI R16,0x00
              OUT DDRD, R16
              OUT PORTD, R16              ; PORTD.2 as Push Button - Input


              LDI R16,0x00         ; Set MCUCR register to enable low level interrupt
              OUT MCUCR, R16


              LDI R16,0x40         ; Set D7 Bit of GICR register to enable interrupt INT1
              OUT GICR, R16

              LDI R16,0x00         ; PORTB as Output
              OUT PORTB, R16

              SEI                         ;

ind_loop:     rjmp ind_loop

int1_ISR:     IN R16, SREG
              PUSH R16

              LDI R16, 0x0A
              MOV R0, R16

c1:           LDI R16, 0x01        ; To blink LED 10 times ( R0 used )
              OUT PORTB, R16       ; Making LED - HIGH

              LDI R16, 0x0C        ; Outer loop - 12 times
a1:           LDI R17, 0xA7        ; Middle loop - 167 times
a2:           LDI R18, 0x7C        ; Inner loop - 124 times
a3:           DEC R18              ; Delay = 1000032 cycles - 0.0032% error
              NOP                  ; Time delay for this loop = 4*a1*a2*a3 + 3*a1*a2 + 3
              BRNE a3
              DEC R17
              BRNE a2
              DEC R16
              BRNE a1

              LDI R16,0x00
              OUT PORTB, R16       ; Making LED - HIGH
              LDI R16, 0x0C        ; 12 times
b1:           LDI R17, 0xA7        ; 167 times
b2:           LDI R18, 0x7C        ; 124 times
b3:           DEC R18
              NOP
              BRNE b3
              DEC R17
              BRNE b2
              DEC R16
              BRNE b1

              DEC R0
              BRNE c1


              POP R16              ; Popping context from Stack
              OUT SREG, R16

              RETI
```

2.1) C program implementing LED blink from **INT1** hardware interrupt

```
#define F_CPU 1000000 // clock frequency

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

ISR(INT1_vect)
{
       int i;
       for (i = 1; i <= 10; i++) // for 10 times LED blink

       {
              PORTB = 0x01;
              // delay of 1 sec = 5*200ms
              for(int j = 1; j<=5; j++)
              {
                     _delay_ms(200) ;
              }
              PORTB = 0x00;
              // delay of 1 sec = 5*200ms
              for(int j = 1; j<=5; j++)
              {
                     _delay_ms(200) ;
              }

       }
}

int main(void)
{
       DDRD = 0x00;         // PORTD as input
       DDRB = 0x01;         // Make PB0 as output
       MCUCR = 0x00;        // Set MCUCR to level triggered
       GICR = 0x80;         // Enable interrupt INT1
       PORTB = 0x00;
       sei();               // global interrupt flag

       while (1)            //wait
       {
       }
}
```

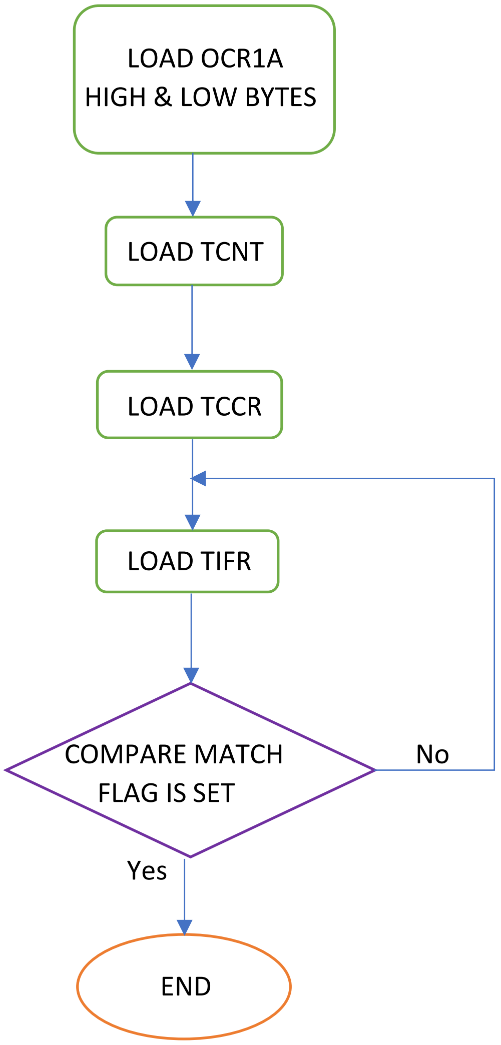
Drive links for above programs:

- INT1_ASM
- INT1_C

II. Blink LED using 16-bit Timer with a duration of 1 second (ON/OFF)

Logic:

a) Flow chart for 1 second delay using 16-bit timer



; OCR1A = 15625 - 1
; For Compare match

; TCNT = 0000 (Initialize)

; TCCR = 0x000B, CTC mode, Pre scaler = 64

; If TIFR flag is set (TCNT1 == OCR1A)

- CTC mode
- Clock frequency = 1MHz => Time period = 1 µs

  Pre scaler = 64 => T = 64 µs

  No of clock cycles required = $\frac{1s}{64\mu s}$ = 15,625

- Load OCR1A with $N - 1$ (Timer value)
- Initialize TCNT register with 0x0000
- Stop delay if OCF1A flag is set

Code:

1) Assembly program for implementing LED blinking using **TIMER1** for 1 second delay and **INT0** interrupt

```
.org 0x0000
rjmp reset

.org 0x0001
rjmp int0_ISR

.org 0x0100
reset:
              ; Loading stack pointer address
              LDI R16,0x70
              OUT SPL, R16
              LDI R16,0x00
              OUT SPH, R16


              LDI R16,0x01         ; Interface port B pin0 to be output
              OUT DDRB, R16        ; So to view LED blinking

              LDI R16,0x00
              OUT DDRD, R16


              LDI R16,0x00         ; Set MCUCR register to enable low level interrupt
              OUT MCUCR, R16


              LDI R16,0x40         ; Set D6 Bit of GICR register to enable interrupt 0 (INT0)
              OUT GICR, R16

              LDI R16,0x00         ; PORTD as input of Push Button signal
              OUT PORTD, R16

              OUT PORTB, R16       ; PORTB as Output - LED

              SEI                  ; Enable Interrupts globally (Break point)

ind_loop:     rjmp ind_loop

int0_ISR:     IN R16,SREG          ; (Break Point)
              PUSH R16

              LDI R16, 0x0C
              OUT PORTD, R16

              LDI R16,0x14
              LDI R17, 0x01
              LDI R18, 0x01

loop:         OUT PORTB, R18                     ; (Break Point)
              EOR R18, R17                       ; Toggle PORTB - LED Output after every sec
              LDI R20, HIGH (15625-1)            ; No of clock cycles = 15625 for 1 sec delay
              OUT OCR1AH, R20                    ;
              LDI R20, LOW (15625-1)
              OUT OCR1AL, R20

              LDI R20,0x00
              OUT TCNT1H, R20                    ; Initialize TCNT1 Registers with zero
              OUT TCNT1L, R20

              OUT TCCR1A, R20
              LDI R20, 0x0B
              OUT TCCR1B, R20                    ; Enable CTC Mode, Pre scaler = 64
again:        IN R20, TIFR
```

```
        SBRS R20, OCF1A                   ; If OCF1A flag is set skip next instruction
        RJMP again

        LDI R20,1<<OCF1A
        OUT TIFR, R20                     ; Clear OCF1A flag

        DEC R16                           ; Loop for next 1 sec delay until 10 blinks
        BRNE loop

        POP R16                           ; Pop contents from stack
        OUT SREG, R16

        RETI                              ; (Break Point)
```

Drive link for above program:

- INT0 TIMER ASM

Inferences:

- Interrupt is more efficient than polling because CPU doesn't need to poll every device that need service. It saves lot of CPU time

- On getting an interrupt, CPU saves the current context in stack registers and jumps to Interrupt Service Routine (ISR) for execution

- Interrupts in AVR will be enabled globally by D7 bit of SREG register using SEI instruction

- Timers can be used to delay time. For large time delays, prescaler can be used

- Loops can be used for delays with appropriate counter values in loop. Nested loops can be used for counter values exceeding the limit of the registers

- Interrupts can be edge/level triggered

- DDR Registers are used to enable Output/Input modes of PORTs

--END---