EE2016 LAB REPORT
AVR INTERRUPT PROGRAMMING

Harish R EE20B044

September 27, 2021

1 Brief Outline

The Aim of this lab session is to implement concepts of *interrupts* and *timers* in Atmel ATmega Microprocessor using AVR Assembly programming and C programming interface. This also requires familiarity in I/O programming in AVR.
The target hardware is ATmega8 chip manufactured by Microchip. The codes are written, compiled and simulated in Microchip Studio.

2 Problem Statements

The following are the problems stated in the experiment :

1. Generate an external (logical) hardware interrupt using an emulation of a push button switch.
2. Write an ISR to blink an LED 10 times with 50% duty cycle. (The lighting of the LED could be verified by monitoring the signal to switch it ON).
3. Write two programs using *INT0* and *INT1* respectively for the external interrupt.
4. Also, one needs to implement all of the above using C-interface.
5. Use the 16 bit timer to make an LED blink with a duration of 1 second. (Optional)

3 Solutions Proposed

3.1 Flowchart

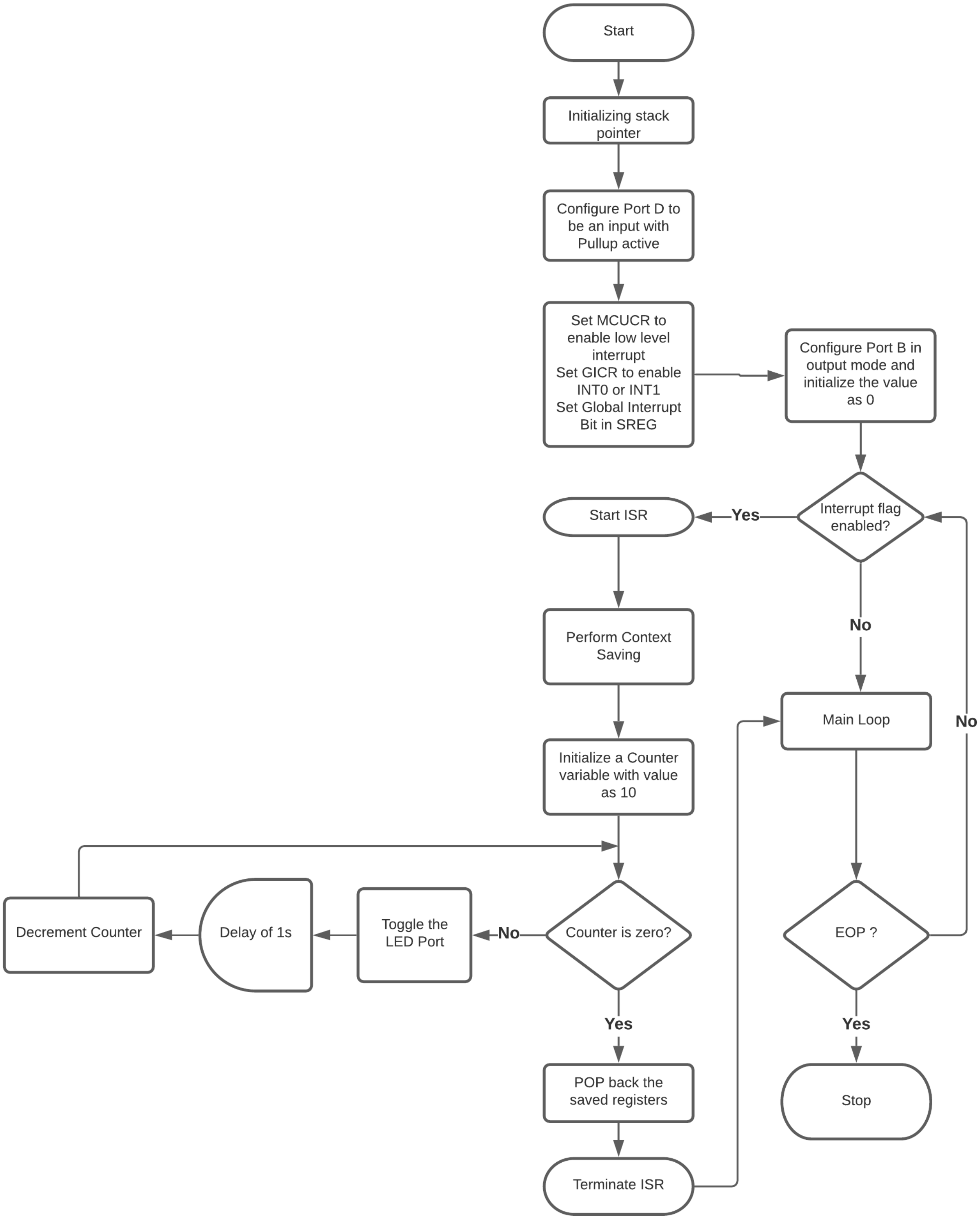
Start
Initializing stack pointer
Configure Port D to be an input with Pullup active
Set MCUCR to enable low level interrupt
Set GICR to enable INT0 or INT1
Set Global Interrupt Bit in SREG
Configure Port B in output mode and initialize the value as 0
Interrupt flag enabled?
Yes
Start ISR
No
Perform Context Saving
Main Loop
No
Initialize a Counter variable with value as 10
Decrement Counter
Delay of 1s
Toggle the LED Port
No
Counter is zero?
EOP ?
Yes
Yes
POP back the saved registers
Stop
Terminate ISR

Figure 1: Flowchart to explain the Logic

Figure 1 shows the logic and flow behind the proposed solutions.

3.2 Logic Explanation

- We first initialize the stack pointer with the address of RAM END.
- Next we configure the pin corresponding to External Hardware Interrupt as Input with Pullup active, to not trigger interrupt by default.
- Set corresponding bits in MCUCR , GICR and SREG to enable low level interrupts and INT1/0, and enable Global Interrupt flag respectively.
- Next We configure the pin corresponding to the LED as output and initialize it with a zero value.
- Now the CPU is kept busy with a infinite while loop.
- To emulate the push of a Switch, The INT pin bit is set. This causes the program to enter the Interrupt Service Routine.
- Now the Function gets executed making the LED blink 10 times with a duty cycle of 50%.
- The Delay is generated by decrementing counters and using nested loops
- Once the ISR is executed, the control returns back to main function by popping the Stack Contents.

3.3 Code

3.3.1 Problem 1 : INT0(Assembly Version)

```
;
; INT0_INTERRUPT.asm
;
; Created: 9/22/2021 3:44:16 PM
; Author : Harish
;



.CSEG

.ORG   0x000
RJMP   RESET

.ORG   0x001
RJMP   INT0_ISR
```

```
.ORG   0x010

RESET:
;Loading the Stack address into SP
    LDI    R16, HIGH(RAMEND)
    OUT    SPH, R16
    LDI    R16, LOW(RAMEND)
    OUT    SPL, R16

;Interface Port B Pin0 to be output
;to control LED blinking
    LDI    R16, (1<<PINB0)
        OUT    DDRB, R16

;Configure PIND2 as Input
;to generate external interrupt INT0
        LDI    R16,     0x00
        OUT    DDRD, R16

;Set ISC1 bits in MCUCR to enable Low level interrupt
    LDI    R16, 0x00
    OUT    MCUCR, R16

;Set INT0 bit in GICR to enable Ext Interrupt 0
    LDI    R16, (1<<INT0)
    OUT    GICR, R16

;Turn off the LED initially
    LDI    R16, 0x00
    OUT    PORTB, R16

;Enable Global Interrupt Flag in SREG
    SEI

;Keep the CPU busy
LOOP:    RJMP    LOOP

INT0_ISR:
;Save the SREG Register
    IN    R16,  SREG
    PUSH  R16

;Make the LED at PINB0 Blink 10 times for time period 2 sec
BLINK:          LDI    R16, 0x0A

;Turn ON the LED at PINB0
        LDI    R16, 0x01
        OUT    PORTB, R16

;Delay for 1 sec
```

```
    RCALL  DELAY_1s

;Turn OFF the LED at PINB0
    LDI    R16, 0x00
    OUT    PORTB, R16

;Delay for 1 sec
    RCALL       DELAY_1s

    DEC         R16
    BRNE    BLINK

;Pop back the SREG
    POP    R16
    OUT    SREG,        R16

    RETI

;SubRoutine to cause 1 sec Delay for CPU of 1 MHz

DELAY_1s:
    LDI    R17, 8

DELAY1:
    LDI    R18, 125

DELAY2:
    LDI    R19, 250

DELAY3:
    DEC    R19
    NOP
    BRNE   DELAY3

    DEC    R18
    BRNE   DELAY2

    DEC    R17
    BRNE   DELAY1

    RET
```

3.3.2 Problem 1 : INT0(C Version)

```
/*
 * INT0_C.c
 *
 * Created: 9/22/2021 4:21:06 PM
 * Author : Harish
 */
```

```
#define         F_CPU  1000000

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


ISR (INT0_vect)
{
        int i;
        for (i=1; i<=10; i++)
        {
                PORTB = 0x01;
                _delay_ms(1000);
                PORTB = 0x00;
                _delay_ms(1000);
        }
}

int main(void)
{
        DDRB = (1<<PINB0);
        DDRD = 0x00;
        MCUCR = 0x00;
        GICR = (1<<INT0);
        PORTB = 0x00;

        sei();

        while (1)
        {
        }
}
```

3.3.3 Problem 2 : INT1(Assembly Version)

```
;
; INT1_Interrupt.asm
;
; Created: 9/22/2021 2:52:50 PM
; Author : Harish
;


.CSEG

.ORG   0x000
RJMP   RESET

.ORG   0x002
```

```
RJMP   INT1_ISR

.ORG   0x010

RESET:
;Loading the Stack address into SP
    LDI    R16, HIGH(RAMEND)
    OUT    SPH, R16
    LDI    R16, LOW(RAMEND)
    OUT    SPL, R16

;Interface Port B Pin0 to be output
;to control LED blinking
    LDI    R16, (1<<PINB0)
        OUT    DDRB, R16

;Configure PIND3 as Input
;to generate external interrupt INT1
        LDI    R16,     0x00
        OUT    DDRD, R16

;Set ISC1 bits in MCUCR to enable Low level interrupt
    LDI    R16, 0x00
    OUT    MCUCR, R16

;Set INT0 bit in GICR to enable Ext Interrupt 1
    LDI    R16, (1<<INT1)
    OUT    GICR, R16

;Turn off the LED initially
    LDI    R16, 0x00
    OUT    PORTB, R16

;Enable Global Interrupt Flag in SREG
    SEI

;Keep the CPU busy
LOOP:    RJMP    LOOP

INT1_ISR:
;Save the SREG Register
    IN    R16,  SREG
    PUSH  R16

;Make the LED at PINB0 Blink 10 times for time period 2 sec
BLINK:          LDI    R16, 0x0A

;Turn ON the LED at PINB0
        LDI    R16, 0x01
        OUT    PORTB, R16
```

```
;Delay for 1 sec
    RCALL  DELAY_1s

;Turn OFF the LED at PINB0
    LDI    R16, 0x00
    OUT    PORTB, R16

;Delay for 1 sec
    RCALL       DELAY_1s

    DEC         R16
    BRNE    BLINK

;Pop back the SREG
    POP    R16
    OUT    SREG,        R16

    RETI

;SubRoutine to cause 1 sec Delay for CPU of 1 MHz

DELAY_1s:
    LDI    R17, 8

DELAY1:
    LDI    R18, 125

DELAY2:
    LDI    R19, 250

DELAY3:
    DEC    R19
    NOP
    BRNE   DELAY3

    DEC    R18
    BRNE   DELAY2

    DEC    R17
    BRNE   DELAY1

    RET
```

3.3.4 Problem 2 : INT1(C Version)

```
/*
 * INT1_C.c
 *
 * Created: 9/22/2021 4:05:47 PM
 * Author : Harish
 */
```

```
#define         F_CPU  1000000

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


ISR (INT1_vect)
{
        int i;
        for (i=1; i<=10; i++)
        {
                PORTB = 0x01;
                _delay_ms(1000);
                PORTB = 0x00;
                _delay_ms(1000);
        }
}

int main(void)
{
    DDRB = (1<<PINB0);
        DDRD = 0x00;
        MCUCR = 0x00;
        GICR = (1<<INT1);
        PORTB = 0x00;

        sei();

    while (1)
    {
    }
}
```

3.3.5 Problem 3 : 16-Bit Timer

```
;
; Timer1_Interrupt.asm
;
; Created: 9/22/2021 4:26:50 PM
; Author : Harish
;


.CSEG

.ORG    0x000

;Initializing the SP with Stack Top address
    LDI    R16, HIGH(RAMEND)
    OUT    SPH, R16
    LDI    R16, LOW(RAMEND)
    OUT    SPL, R16

;Configure PINB0 in output mode
    LDI    R16, (1<<PINB0)
    OUT    DDRB, R16

;Blink LED in PINB0 with Time period 2 sec
BEGIN:
    SBI    PORTB, 0
    RCALL     DELAY_1s
    CBI    PORTB, 0
    RCALL     DELAY_1s

    RJMP    BEGIN

;------------TIMER1 DELAY----------------------

DELAY_1s:
    LDI    R20, 0x00
    OUT    TCNT1H, R20
    OUT    TCNT1L, R20

;Loading 15625 into OCR1A
    LDI    R20, HIGH(0x3D09)
    OUT    OCR1AH, R20
    LDI    R20, LOW(0x3D09)
    OUT    OCR1AL, R20

;Setting Prescalar as 1/64
    LDI    R20, 0x03
    OUT    TCCR1B, R20

;Setting CTC Mode for Channel A
    LDI    R20, (1<<6)
    OUT    TCCR1A, R20

AGAIN:
    IN     R20, TIFR
    SBRS   R20, OCF1A
    RJMP    AGAIN

    LDI    R20, 1<<OCF1A
    OUT    TIFR, R20
    LDI    R19, 0
    OUT    TCCR1B, R19
    OUT    TCCR1A, R19

    RET
```

4 Inferences

The below are my inferences and personal learnings from the experiment :

- Proper Debugging in Microchip Studio and Usage of breakpoints to analyze code segments.
- Advantages of using an Interrupt mechanism over Polling to keep track of external events.
- Learnt the usage of 16 bit timers effectively to create accurate delays.
- Converting back and forth into C - interface and Assembly programming to a certain extent.