Experiment 2: Interrupts and Timers in Atmel AVR Atmega

Aim –

Using Atmel AVR assembly language programming, implement interrupts and timers in Atmel Atmega microprocessor. The main constraint is that, it should be emulation only (due to ongoing pandemic). Aims of this experiment are
(i) Generate an external (logical) hardware interrupt using an emulation of a push button switch.
(ii) Write an ISR to switch ON an LED for a few seconds (10 secs) and then switch OFF. (The lighting of the LED could be verified by monitoring the signal to switch it ON).

Questions –

Use int0 to redo the same in the demo program (duely filled in). Once the switch is pressed the LED should blink 10 times (ON (or OFF) - 1 sec, duty cycle could be 50 %). Demonstrate both the cases.

Rewrite the program in 'C' (int1). Rewrite the C program for int0.

CODE-

For Interupt int0
Assembly code-

```
;
; Lab02_Interupt_Int0.asm
;
; Created: 25-09-2021 19:23:45
; Author : suhas
;

#include "m8def.inc"

.org 0
rjmp reset ;

.org 0x0000;
rjmp int0_ISR;

.org 0x0100;

reset:
        LDI R16,0x70 ;
        OUT SPL,R16 ;
        LDI R16,0x00;
        OUT SPH,R16;

        LDI R16,0x01;
        OUT DDRB,R16;

        LDI R16,0x00;
        OUT DDRD,R16;

        IN R16, MCUCR;
        ORI R16, 0x00;
        OUT MCUCR, R16;

        IN R16, GICR;
        ORI R16, 0x40;
        OUT GICR, R16;

        LDI R16, 0x00;
        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;
    OUT PORTB, R16

    LDI R16, 0xFF
    a1:         LDI R17, 0xFF
    a2:         DEC R17
    BRNE a2
    DEC R16
    BRNE a1

    LDI R16, 0x00
    OUT PORTB, R16

    LDI R16, 0xFF
    b1:         LDI R17, 0xFF
    b2:         DEC R17
    BRNE b2
    DEC R16
    BRNE b1

    DEC R0
    BRNE c1
    POP R16
    OUT SREG, R16

    RETI
```

C programme-

```
#define F_CPU 1000000

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

ISR (INT0_vect)
{

        for(int i=0; i<10; i=i+1)
        {
                //PortB is set to 1 for 1 sec (ON State)
                PORTB = 0x01;
                _delay_ms(1000);
                //PortB is set to 0 for 1 sec (ON State)
                PORTB = 0x00;
                _delay_ms(1000);
        }
}

int main (void)
{
        //port i/o declarations
        DDRD = 0x00;
        DDRB = 0x01;
        MCUCR = 0x00;
        GICR = 0x40;
        PORTB = 0x00;

        //set interrupt flag of SREG
        sei();

        while (1)
        {
                //To keep the program running forever.
        }
}
```

For Interupt int1
Assembly code-

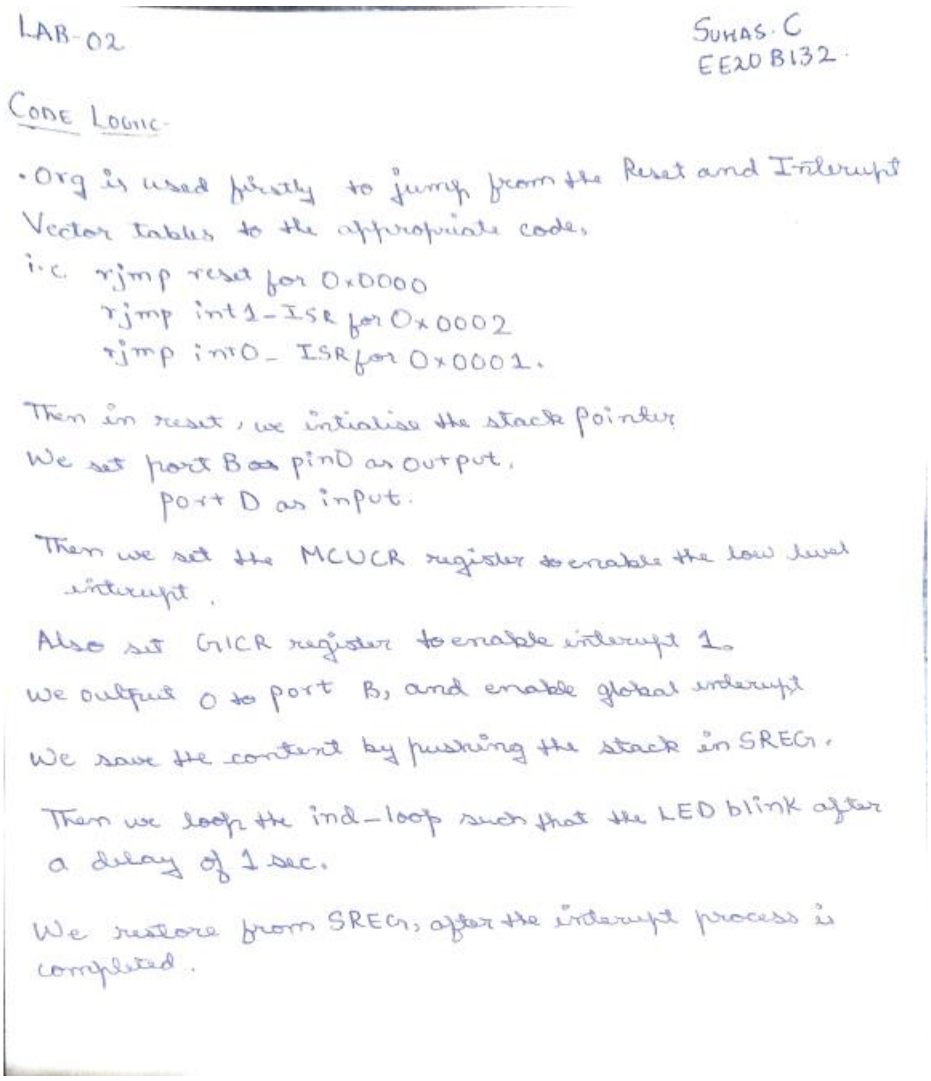
```
;
; LAB_02_Interupt_int1.asm
;
; Created: 26-09-2021 02:17:45
; Author : suhas
;

#include "m8def.inc"

.org 0
rjmp reset ;

.org 0x0000;
rjmp int1_ISR;

.org 0x0100;

reset:
        LDI R16,0x70 ;
        OUT SPL,R16 ;
        LDI R16,0x00;
        OUT SPH,R16;

        LDI R16,0x01;
        OUT DDRB,R16;
        LDI R16,0x00;
        OUT DDRD,R16;

        IN R16, MCUCR;
        ORI R16, 0x00;
        OUT MCUCR, R16;

        IN R16, GICR;
        ORI R16, 0x80;
        OUT GICR, R16;

        LDI R16, 0x00;
        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;
    OUT PORTB, R16

    LDI R16, 0xFF
    a1:         LDI R17, 0xFF
    a2:         DEC R17
    BRNE a2
    DEC R16
    BRNE a1

    LDI R16, 0x00
    OUT PORTB, R16

    LDI R16, 0xFF
    b1:         LDI R17, 0xFF
    b2:         DEC R17
    BRNE b2
    DEC R16
    BRNE b1

    DEC R0
    BRNE c1
    POP R16
    OUT SREG, R16

    RETI
```

C programme-

```
#define F_CPU 1000000

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

```
ISR (INT1_vect)
{

        for(int i=0; i<10; i=i+1)
        {
                //PortB is set to 1 for 1 sec (ON State)
                PORTB = 0x01;
                _delay_ms(1000);
                //PortB is set to 0 for 1 sec (ON State)
                PORTB = 0x00;
                _delay_ms(1000);
        }
}

int main (void)
{
        //port i/o declarations
        DDRD = 0x00;
        DDRB = 0x01;
        MCUCR = 0x00;
        GICR = 0x80;
        PORTB = 0x00;

        //set interrupt flag of SREG
        sei();

        while (1)
        {
                //To keep the program running forever.
        }
}
```

CODE LOGIC –



LAB-02

Suhas. C
EE20B132.

Code Logic-

• Org is used firstly to jump from the Reset and Interrupt Vector tables to the appropriate code,

i.e. rjmp reset for 0x0000
 rjmp int1-ISR for 0x0002
 rjmp int0_ ISR for 0x0001.

Then in reset, we initialise the stack pointer

We set port B as pinD as output.
 port D as input.

Then we set the MCUCR register to enable the low level interrupt.

Also set GICR register to enable interrupt 1.

We output 0 to port B, and enable global interrupt

We save the content by pushing the stack in SREG.

Then we loop the ind-loop such that the LED blink after a delay of 1 sec.

We restore from SREG, after the interrupt process is completed.

Learning from The Experiment –

1. Learned about the Loops, Timers and Interrupts in the AVR Atmega.
2. Learning to code in C
3. Running the C code in the Microchip.