EE2016 Experiment-6

Group-3 EE23B027, EE23B033, EE23b039

Task 1

To connect the press button switch, one LED that gives white light and another that gives red light

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include<util/delay.h>
#include<avr/interrupt.h>

  int main(void)
{
 DDRB=0x03;
 DDRD=0x00;
GICR=0x40;
 SREG=0x80;
while(1) {
 PORTB=0x01;
}
}

 ISR(INT0_vect) {
 cli();
  PORTB=0x02;
 _delay_ms(100);
PORTB=0x00;
 _delay_ms(100);
 sei();
}
```

Debugging

The led start blinking if the button is pressed otherwise the white will glow
Click here to see the execution of Task 1

Task 2

To cause a (red) LED to blink upon receiving an interrupt via a button press

```
.org 0
rjmp reset ; on reset, program starts here

.org 0x002 ; Interrupt vector address for INT1.
rjmp int1_ISR ; Jump to INT1 interrupt service routine

reset:
    ldi R16, 0x70 ; setup the stack pointer to point to address 0x0070
    out SPL, R16 ; Set stack pointer low byte
    ldi R16, 0x00 ; Stack pointer high byte
    out SPH, R16 ; Set stack pointer high byte

    ldi R16, 0x02 ; Make PB1 (PORTB, pin 1) output
    out DDRB, R16 ; Set data direction for PORTB

    ldi R16, 0x00 ; Make PORTD input
    out DDRD, R16 ; Set data direction for PORTD

    ldi R16, 0x08 ; Enable pull-up resistor on PD3
    out PORTD, R16 ; Set PORTD register
```

```
    in R16, GICR ; Read SREG
    ori R16, 0x80 ; Enable INT1 interrupt (set the I-bit)
    out GICR, R16 ; Write back to SREG

    ldi R16, 0x00 ; Turn off LED (assuming it's connected to PB1)
    out PORTB, R16 ; Write to PORTB

    sei ; Enable global interrupts

indefiniteloop:
    rjmp indefiniteloop ; Infinite loop

int1_ISR: ; INT1 interrupt handler or ISR
cli ;
    in R16, SREG ; Save status register SREG
    push R16 ; Push SREG onto the stack

    ldi R16, 10 ; Blink LED 10 times
    mov R0, R16 ; Move count into R0

back5:
    ldi R16, 0x02 ; Turn on LED (PB1)
    out PORTB, R16 ; Set PORTB, pin 1 high

delay1:
    ldi R16, 0xFF ; Delay value
back2:
LDI R17, 0xFF

back1:
    dec R17 ; Decrement delay
    brne back1 ; Repeat until zero
    dec R16
    brne back2

    ldi R16, 0x00 ; Turn off LED (PB1)
    out PORTB, R16 ; Set PORTB, pin 1 low

delay2:
    ldi R16, 0xFF ; Delay value

back3:
    ldi R17, 0xFF
back4:
    dec R17 ; Decrement delay
    brne back4 ; Repeat until zero

    dec R16
    brne back3

    dec R0 ; Decrement blink counter
    brne back5 ; If not zero, blink again

    pop R16 ; Retrieve status register
    out SREG, R16 ; Restore status register
    sei
    reti ; Return from interrupt
```

Debugging

The red led will blink 10 times if interrupt is given.

Task 3

To include a white LED to Task 2

```
.org 0
```

```
rjmp reset ; On reset, program starts here

.org 0x002 ; Interrupt vector address for INT1.
rjmp int1_ISR ; Jump to INT1 interrupt service routine

reset:
    ldi R16, 0x70 ; Setup the stack pointer to point to address 0x0070
    out SPL, R16 ; Set stack pointer low byte
    ldi R16, 0x00 ; Stack pointer high byte
    out SPH, R16 ; Set stack pointer high byte

    ldi R16, 0x03 ; Make PB0 (White LED) and PB1 (Red LED) outputs
    out DDRB, R16 ; Set data direction for PORTB

    ldi R16, 0x00 ; Make PORTD input
    out DDRD, R16 ; Set data direction for PORTD

    ldi R16, 0x08 ; Enable pull-up resistor on PD3
    out PORTD, R16 ; Set PORTD register

    in R16, GICR ; Read GICR
    ori R16, 0x80 ; Enable INT1 interrupt (set the I-bit)
    out GICR, R16 ; Write back to GICR

    ldi R16, 0x00 ; Turn off both LEDs initially
    out PORTB, R16 ; Set both PB0 and PB1 low

    sei ; Enable global interrupts

indefiniteloop:
    rjmp indefiniteloop ; Infinite loop

; Interrupt Service Routine (ISR) for INT1
int1_ISR:
    cli ; Disable interrupts

    in R16, SREG ; Save status register SREG
    push R16 ; Push SREG onto the stack

    ldi R16, 10 ; Blink red LED (PB1) 10 times
    mov R0, R16 ; Move count into R0

back5:
    ldi R16, 0x02 ; Turn on red LED (PB1)
    out PORTB, R16 ; Set PB1 high

delay1:
    ldi R16, 0xFF ; Delay value
back2:
    ldi R17, 0xFF
back1:
    dec R17 ; Decrement delay
    brne back1 ; Repeat until zero
    dec R16
    brne back2 ; Repeat delay loop

    ldi R16, 0x00 ; Turn off both LEDs (PB0 and PB1)
    out PORTB, R16 ; Set both low

delay2:
    ldi R16, 0xFF ; Delay value
back3:
    ldi R17, 0xFF
back4:
    dec R17 ; Decrement delay
    brne back4 ; Repeat until zero
    dec R16
    brne back3 ; Repeat delay loop

    dec R0 ; Decrement blink counter
```

```
brne back5 ; If not zero, blink again

ldi R16, 0x01 ; After blinking, turn on the white LED (PB0)
out PORTB, R16 ; Set PB0 high

pop R16 ; Retrieve status register
out SREG, R16 ; Restore status register

sei ; Re-enable interrupts
reti ; Return from interrupt
```

Debugging

The red led will blink 10 times on interrupt then the white led will turn on
Click here to see the execution of Task 3

Error and Debugging

While execution of the third task the white led and red led were alternatively blinking instead of giving 10 blinks of red led then turning on white. We rectified it by change our code to make the white led blink after the red led completes 10 blinks.