Experiment 4: Interrupts in Atmel AVR Atmega through Assembly Programming

Santhosh S P ee21b119

1 Task 1: Code for making the LED blink using INT1

```
    .org 0
rjmp reset

.org 0x0002
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;Load MCUCR register
  ORI R16,0x80
  OUT MCUCR,R16

  IN R16,GICR; Load GICR register
  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 R16,0x02
 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
```

2 Task 2: Using INT0 for the above task

```
   .org 0
rjmp reset

.org 0x0001
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;Load MCUCR register
   ORI R16,0x80
   OUT MCUCR,R16

   IN R16,GICR; Load GICR register
   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 R16,0x02
  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
```

3 Task 3: Rewriting using C

3.1 C code to implement INT1:

```
//defining the CPU clock frequency
#define F_CPU 1000000

//including the required header files
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

ISR (INT1_vect)
{
for (j=0;j<10;j=j+1)
        //making the led blink 10 times
{
            //makes the last bit in PORTB high
     PORTB=0x01;
            //stays there for a second
     _delay_ms(1000);
            //makes the last bit in PORTB low
```

```
     PORTB=0x00;
            //stays there for a second
     _delay_ms(1000);
}


}
int main(void)
{

DDRD=0x00;

        //Set PB0 to output
DDRB=0x00;

        //Set MCU Control Register to level triggered
MCUCR=0x80;

        //Enabling INT1
GICR=0x00;

PORTB=0x00;
sei();
        // global interrupt flag

        //setting an infinite loop
while (1)
{

}
}
```

3.2 C code to implement INT0:

```
//defining the clock frequency
#define F_CPU 1000000

//including the required header files
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

ISR (INT0_vect)
{
    //making the led blink 10 times
    for(int i=0; i<10; i=i+1)
```

```
    {
        //makes the last bit in PORTB high
        PORTB = 0x01;
        //waits for a second
        _delay_ms(1000);
        //makes the last bit in PORTB low
        PORTB = 0x00;
        //waits for a second
        _delay_ms(1000);
    }
}

int main (void)
{
    //declaring i/o configuration of ports
    DDRD = 0x00;
    DDRB = 0x01;
    MCUCR = 0x02;
    GICR = 0x40;
    PORTB = 0x00;

    sei();
    //global interrupt flag

    //infinite loop
    while (1)
    {

    }
}
```

4 Task 4: All demonstrations and outputs

4.1 Outputs for INT1:

Video: Making the LED blink ten times using INT1

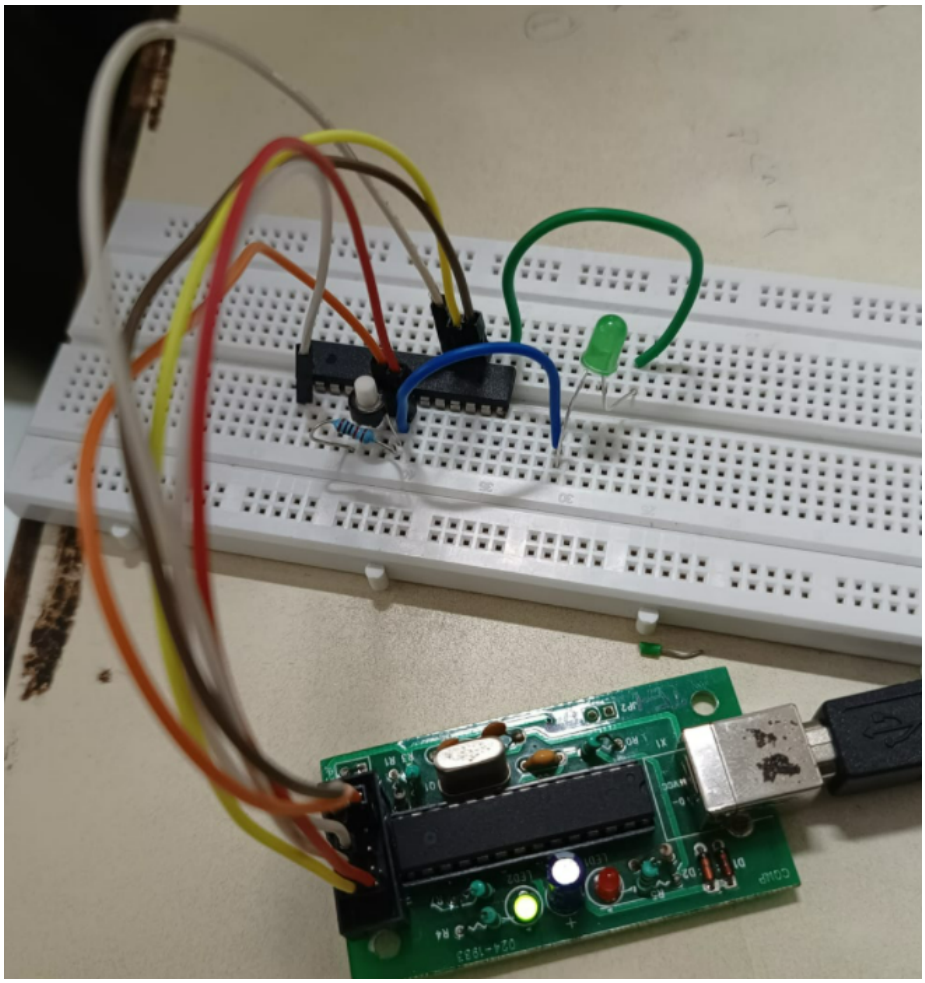

Figure 1: Circuit to implement int1

4.2 Outputs for INT0:

Video: Making the LED blink ten times using INT0

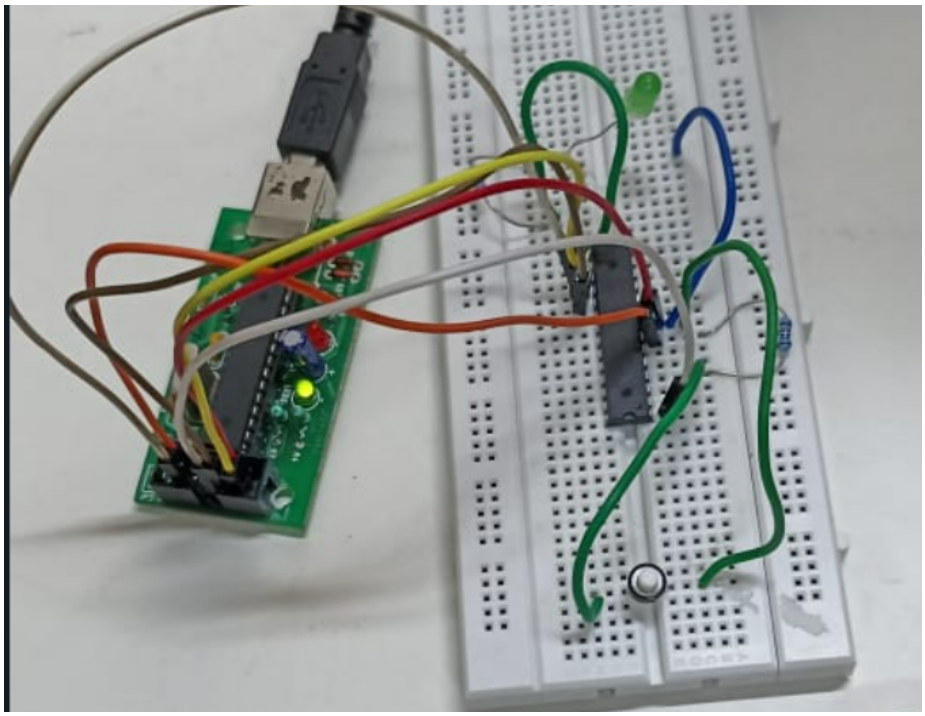

Figure 2: Circuit to implement int0