EE2016: Microprocessors Lab
Experiment # 4: Interrupts in Atmel AVR through Assembly Programming

Amizhthni PRK, EE21B015
Nachiket Dighe, EE21B093

October 15, 2022

Contents

List of Figures

1 Assembly

1.1 Aim:

Using Atmel AVR assembly language programming:

1. Generate an external (logical) hardware interrupt using an emulation of a push button switch.
2. 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).

1.2 Procedure

- Set the corresponding pins as input and output.
- Wire the circuit as shown in figure.
- Write code in AVR aseembly language to emulate the push button so that the LED blinks for 10secs.

1.3 Circuit diagram

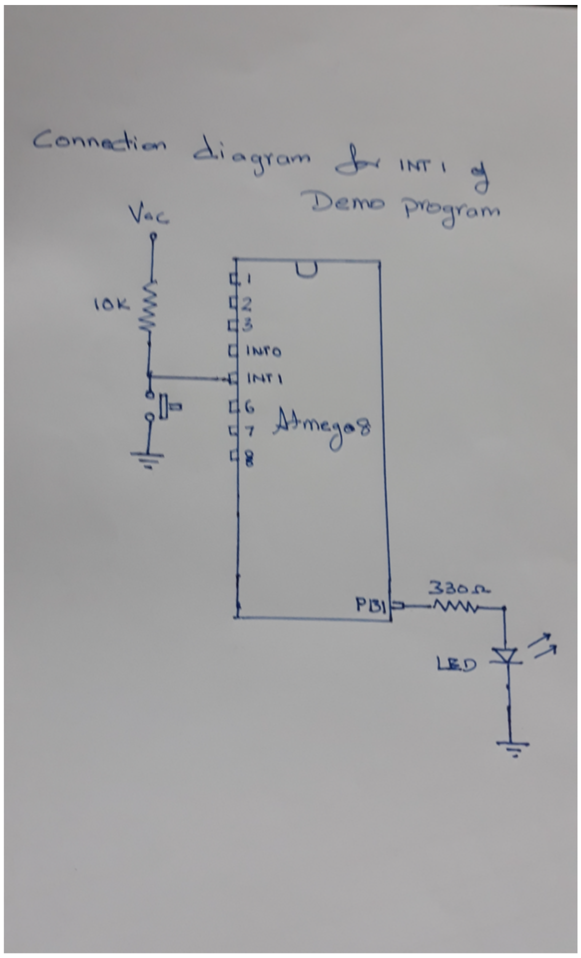

Figure 1: Output circuit board

1.4 Code

1.4.1 Using Interrupt int0

The AVR assembly code for controlling the LED and making it Blink for 10secs using **int0** Interrupt is shown below in linking 1:

```
.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

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

        LDI R16,0x00
        OUT DDRD,R16

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

        ;Set GICR register to enable interrupt 1
        LDI R16, 1<<INT0
        OUT GICR,R16

        LDI R16,0x00
        OUT PORTB,R16

        SEI
ind_loop:
        rjmp ind_loop





;Interupt Service Routine

```

```
int0_ISR:
        IN R16,SREG
        PUSH R16

        LDI R16,0x0A ;For blinking LED 10 times
        MOV R0,R16

c1:        LDI R16, 0x01;Setting LED to High (ON)
        OUT PORTB, R16

;Program To delay
;start
        LDI R16, 200
a1:         LDI R17, 100
a2:         LDI R18, 10
a3:        NOP                ;to waste cycle
        NOP

        DEC R18
        BRNE a3

        DEC R17
        BRNE a2

        DEC R16
        BRNE a1

        LDI R16, 0x00  ;Setting LED to Low (Off)
        OUT PORTB, R16
;end
;Program To delay
;start
        LDI R16, 200
b1:         LDI R17, 100
b2:         LDI R18, 10
b3:         NOP                ;to waste cycle
        NOP

        DEC R18
        BRNE b3

        DEC R17
        BRNE b2

        DEC R16
        BRNE b1
;end

        DEC R0
        BRNE c1 ;To repeat blinking 10 times

        POP R16
        OUT SREG, R16 ;restoring values before interruption.

        RETI
```

Listing 1: Using int0 Interrupt in AVR

1.4.2 Using Interrupt int1

The AVR assembly code for controlling the LED and making it Blink for 10secs using **int1** Interrupt is shown below in linking 2:

```
.org 0x0000
rjmp reset

.org 0x0002
rjmp int1_ISR

.org 0x0100

reset:
        ;Loading stack pointer address

        LDI R16,0x70
        OUT SPL,R16
        LDI R16,0x00
        OUT SPH,R16

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

        LDI R16,0x00
        OUT DDRD,R16

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

        ;Set GICR register to enable interrupt 1
        LDI R16, 1<<INT1
        OUT GICR,R16

        LDI R16,0x00
        OUT PORTB,R16

        SEI
ind_loop:
        rjmp ind_loop





;Interupt Service Routine

int1_ISR:
        IN R16,SREG
        PUSH R16

        LDI R16,0x0A ;For blinking LED 10 times
        MOV R0,R16

c1:        LDI R16, 0x01;Setting LED to High (ON)
        OUT PORTB, R16

;Program To delay
;start
        LDI R16, 200
a1:         LDI R17, 100
a2:         LDI R18, 10
a3:        NOP                ;to waste cycle
        NOP

        DEC R18
        BRNE a3

        DEC R17
        BRNE a2

        DEC R16
        BRNE a1

        LDI R16, 0x00  ;Setting LED to Low (Off)
        OUT PORTB, R16
;end
;Program To delay
;start
        LDI R16, 200
b1:         LDI R17, 100
b2:         LDI R18, 10
b3:         NOP                ;to waste cycle
        NOP

        DEC R18
        BRNE b3

        DEC R17
        BRNE b2

        DEC R16
        BRNE b1
;end

        DEC R0
        BRNE c1 ;To repeat blinking 10 times

        POP R16
        OUT SREG, R16 ;restoring values before interruption.

        RETI
```

Listing 2: Using int1 Interrupt in AVR

1.5 Input and Output

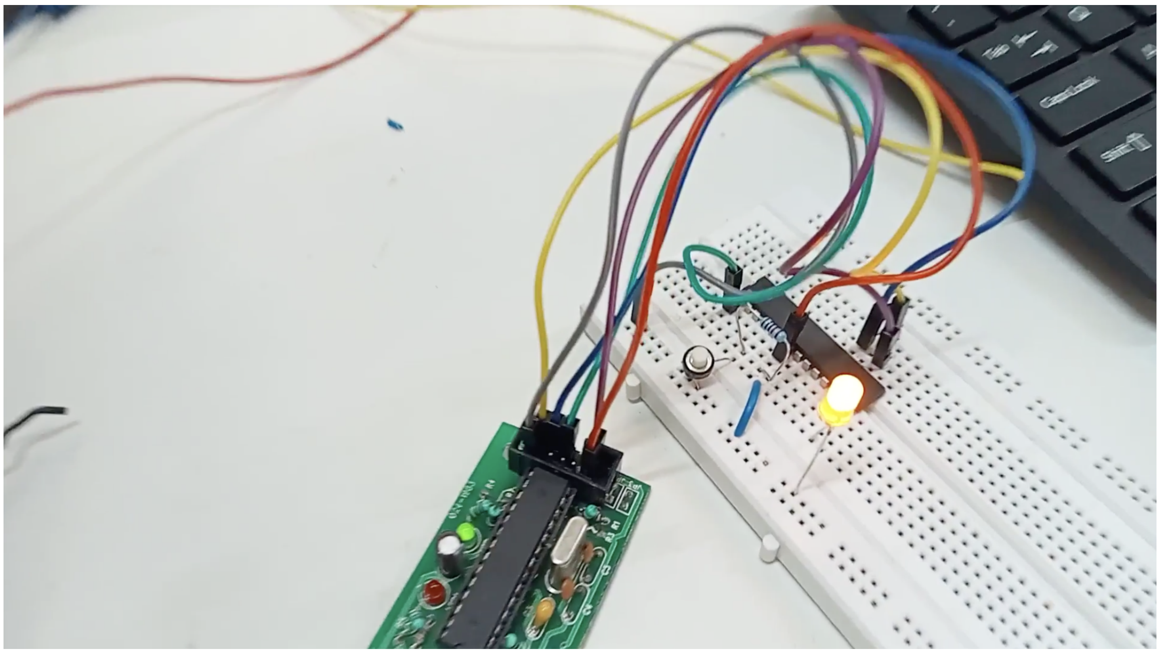

Figure 2: Output of LED blinking

2 C code

2.1 Aim:

Using C language programming:

1. Generate an external (logical) hardware interrupt using an emulation of a push button switch.
2. 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).

2.2 Procedure

- Set the corresponding pins as input and output.
- Wire the circuit as shown in figure.
- Write code in C language to emulate the push button so that the LED blinks for 10secs.

2.3 Code

2.3.1 Using Interrupt int0

The C language code for controlling the LED and making it Blink for 10secs using Interrupt int0 is shown in linking 3:

```c
#define F_CPU 1000000
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

ISR (INT1_vect)
{
        // ISR to blink the LED 10 times
        // ON and OFF interval of 1 second each
        int i;
        for(i=0; i<10; i++)
        {
                //PB0 is set to High for 1 sec.
                PORTB = 0x01;
                _delay_ms(1000);//Including delay of 1 sec

                //PB0 is set to Low for 1 sec.
                PORTB = 0x00;
                _delay_ms(1000);//Including delay of 1 sec
        }
}

int main (void)
{
        DDRD = 0x00;
        DDRB = 0x01; //Make PB0 as output

        MCUCR = 0x00; //Set MCUCR to level triggered

        GICR = (1<<INT0); //Enable interrupt 0

        PORTB = 0x00;

        sei(); // global interrupt flag
        while (1)
        {
                //To keep the program running forever.
        }
}
```

Listing 3: Using Interrupt int0 in C

2.3.2 Using Interrupt int1

The C language code for controlling the LED and making it Blink for 10secs using Interrupt int1 is shown in linking 4:

```
#define F_CPU 1000000
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

ISR (INT1_vect)
{
        // ISR to blink the LED 10 times
        // ON and OFF interval of 1 second each
        int i;
        for(i=0; i<10; i++)
        {
                //PB0 is set to High for 1 sec.
                PORTB = 0x01;
                _delay_ms(1000);//Including delay of 1 sec

                //PB0 is set to Low for 1 sec.
                PORTB = 0x00;
                _delay_ms(1000);//Including delay of 1 sec
        }
}

int main (void)
{
        DDRD = 0x00;
        DDRB = 0x01; //Make PB0 as output

        MCUCR = 0x00; //Set MCUCR to level triggered

        GICR = (1<<INT1); //Enable interrupt 1

        PORTB = 0x00;

        sei(); // global interrupt flag
        while (1)
        {
                //To keep the program running forever.
        }
}
```

Listing 4: Using Interrupt int1 in C

3 Explanation

1. The external interrupts are triggered by the INT0, and INT1 pins. Observe that, if enabled, the interrupts will trigger even if the INT0 or 1 pins are configured as outputs.

2. This feature provides a way of generating a software interrupt. The external interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU Control Register MCUCR.

3. When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 and INT1 requires the presence of an I/O clock.

4. Low level interrupts on INT0/INT1 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

4 Videos and Outputs

The Output Video of the LED blinking using Interrupts is given in the Video Link