Experiment 4: Interrupts in Atmel AVR Atmega through Assembly Programming

Priyanka Umasankar, EE23B060

Objective

- Using Atmel AVR assembly language programming, implement interrupts and DIP switches control in Atmel Atmega microprocessor. (i) Generate an external (logical) hardware interrupt using an emulation of a push button switch. (ii) Write an ISR (Interrupt Service Routine) 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).

Equipments and Hardware Required

1. Atmel Atmega8 Microcontroller chip, USBASP programmer,
2. Bread board with hardware components, data / power cables, LED, mini push buttons etc
3. A PC with Microchip/ Studio simulation software loaded and AVR Burn-o-mat software

1 Procedure

This assembly program for the Atmel Atmega8 microcontroller handles an external interrupt (`INT1`) and blinks an LED connected to Port B pin 0. Below is the summary of the key operations:

- Stack Initialization: The stack pointer is initialized to `0x70` to manage interrupts and subroutine calls.
- Port Configuration: Port B pin 0 is set as an output to control the LED, and Port D is set as input.
- Interrupt Setup:
  - The MCU Control Register (`MCUCR`) is configured to trigger an external interrupt on `INT1` on a rising edge.

  - The General Interrupt Control Register (`GICR`) enables the `INT1` interrupt.

- Timer1 Configuration: - Timer1 is set in normal mode with a prescaler of 1024, with the `OCR1A` register configured for a 1-second delay.
- Main Loop: The program remains in an infinite loop, waiting for an interrupt event to occur.
- Interrupt Service Routine (ISR):
  - Upon interrupt, the ISR disables global interrupts and blinks the LED 10 times using a loop. Each blink includes turning the LED on, adding a delay using nested decrement loops, and then turning the LED off.
  - Once the LED has blinked 10 times, global interrupts are re-enabled, and the program returns from the interrupt.

1.1 Code

Listing 1: Interrupts in Atmel AVR Atmega through Assembly Programming

```
.org 0
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 R16,0x02
 OUT DDRB,R16
 LDI R16,0x00
 OUT DDRD,R16

 ;Set MCUCR register to enable low level interrupt
 LDI R16, (1 << ISC11) | (1 << ISC10)
 OUT MCUCR,R16

```

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


      ; Set Timer1 for 1 second delay
      LDI R16, 0x00
      OUT TCCR1A, R16    ; Normal mode
      LDI R16, (1 << CS12) | (1 << CS10)  ; Prescaler = 1024
      OUT TCCR1B, R16
      LDI R16, 0x00
      OUT TCNT1H, R16
      LDI R16, 0x00
      OUT TCNT1L, R16
      LDI R16, 0xF9    ; 1 second delay value for TCNT1
      OUT OCR1AH, R16
      LDI R16, 0xF9
      OUT OCR1AL, R16

      ; Enable Timer1 Compare Match interrupt
      LDI R16, (1 << OCIE1A)
      OUT TIMSK, R16



 LDI R16,0x00
 OUT PORTB,R16

 SEI
ind_loop:rjmp ind_loop

int1_ISR:
      ; Disable global interrupts during ISR
      CLI

      ; LED Blink 10 times
      LDI R16, 10        ; Number of blinks
      MOV R0, R16

blink_loop:
      LDI R16, 0x02
      OUT PORTB, R16    ; Turn LED ON
      LDI R16, 0xFF
a1:  LDI R17, 0xFF
a2:  DEC R17
      BRNE a2
      DEC R16
      BRNE a1

      LDI R16, 0x00
```

```
78        OUT PORTB, R16     ; Turn LED OFF
79        LDI R16, 0xFF
80  b1:   LDI R17, 0xFF
81  b2:   DEC R17
82        BRNE b2
83        DEC R16
84        BRNE b1
85
86        DEC R0
87        BRNE blink_loop
88
89
90
91        ; Re-enable global interrupts
92        SEI
93
94        ; Clear the interrupt flag and return from ISR
95        RETI
```

1.2 Result

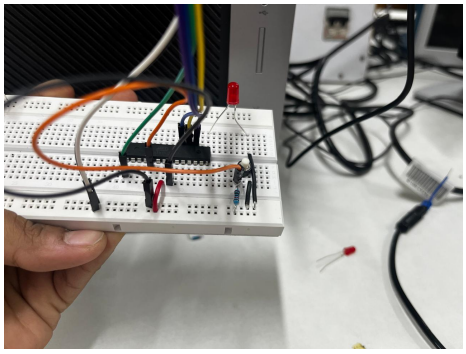

Figure 1: switch ON an LED for a 10 seconds and then switch OFF

1.3 Demonstration Video

Video Link