EE2016 : Microprocessor Theory and Lab

Lab Report # 4
INTERRUPTS IN ATMEL AVR ATMEGA THROUGH ASSEMBLY PROGRAMMING

Mandla Siva Manoj, EE21B083
Ajoe George, EE21B009
Group 3

October 15, 2022

Contents

List of Figures

1 Aim of the Experiment

Using Atmel AVR assembly language programming, implement interrupts and DIP switches control in Atmel Atmega microprocessor. Aims of this experiment are:

- Generate an external (logical) hardware interrupt using an emulation of a push button switch.
- 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).
- If there is time, you could try this also: Use the 16 bit timer to make an LED blink with a duration of 1 second.

2 Problems

1. Fill in the blanks in the assembly code.
2. Use int0 to redo the same in the demo program (duly 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.
3. Rewrite the program in 'C' (int1). Rewrite the C program for int0.
4. Demonstrate both the cases (of assembly and C).

3 INT 0

3.1 Code in Assembly Language

```
//ASM program to implement external input INT0 in AVR
//switch to PD2
//LED to PB0

.nolist
.include "m8def.inc"
.list

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

.org 0x0002     ; Interrupt vector address for INT0. Put your ISR here or
↪   jump
rjmp INT0_ISR   ; to the ISR

.org 0x0100

reset:
     ldi R16,0x70 ; Setup the stack pointer to point to address 0x0070
         out spl,R16
         ldi R16,0x00
         out sph,R16

         ldi R16,0x01     ;make PB0 output
         out DDRB,R16

         ldi R16,0x00    ; make PORTD as input
         out DDRD, R16

         ldi R16,0x04    ; use pull up resistor for PD2
         out PORTD,R16

         in R16,MCUCR
         ori R16,0x02     ; set INT0 to falling edge sensitive
         out MCUCR,R16    ; use OR so that other bits are not affected

         in R16,GICR
         ori R16, 0x40    ; enable INT0 interrupt
         out GICR,R16

         ldi R16,0x00     ; Turn off LED
         out PORTB,R16

         sei               ; enable interrupts

         indefiniteloop: rjmp indefiniteloop

         INT0_ISR:             ; INT0 Interrupt handler or ISR
                in R16,SREG    ; save status register
                        push R16

                        ldi R16,0x0a   ; blink LED 10 times
                        mov R0,R16

        c1: ldi R16,0x01    ;Turn ON LED
            out PORTB,R16

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

                 ldi R16,0x00   ; Turn OFF LED
                 out PORTB,R16

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

                 DEC R0
                 BRNE c1    ; check if LED has blinked 10 times

                 pop  R16      ; retrive status register
                 out SREG, R16

                 RETI           ; go back to main program
```

Listing 1: Code for INT0 in ASM

3.2 Code in C Language

```
//C program for INT0
#define F_CPU 1000000
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

ISR (INT0_vect)
{
        // Write your ISR here to blink the LED 10 times
        // with ON and OFF interval of 1 second each
        for(int i=0; i<10; i=i+1)
```

```c
        {
                //PB0 is set to 1 for 1 sec.
                PORTB = 0x01;
                _delay_ms(1000);
                //PB0 is set to 0 for 1 sec.
                PORTB = 0x00;
                _delay_ms(1000);
        }
}

int main (void)
{
        //port i/o declarations
        DDRD = 0x00;
        DDRB = 0x01;
        MCUCR = 0x02; //check
        GICR = 0x40;
        PORTB = 0x00;

        //set interrupt flag of SREG
        sei();

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

Listing 2: Code for INT0 in C

3.3 Outputs

Video for INT0 ASM:
https://drive.google.com/file/d/1GmlwjWGwj5UysEHztOQ6WA6dIoC6APzz/view?usp=drivesdk

Video for INT0 C:
https://drive.google.com/file/d/1GePhhQhB-NP2EgTQnhWwiN2enPLHKmlh/view?usp=drivesdk

3.4 Explanation

- The INT0 interrupt, which is associated with pin 4 on an ATmega8, detects various levels and level changes on the INT0 input.
- The interrupt branches to the INT0 vector if the corresponding interrupt is enabled. The states or changes that cause this interrupt can be chosen using the bits ISC01 and ISC00.

- Determining whether the counter is at zero is the main goal. The T flag is checked if it does. If set, the counter is reset to six and a new cycle begins.
- The bTo flag is checked if the cycle counter is not zero (it is set by the CTC ISR when a time-out occurs). If this is set, the next stage is handled. The flow returns to the sleep instruction in any case.

4 INT 1

4.1 Code in Assembly Language

```
//ASM program to implement external input INT1 in AVR
//switch to PD3
//LED to PB0

.nolist
.include "m8def.inc"
.list

.org 0
rjmp reset      ; on reset, program starts here
.org 0x0002     ; Interrupt vector address for INT1. Put your ISR here or
↪   jump
rjmp INT1_ISR   ; to the ISR

.org 0x0100

reset:
     ldi R16,0x70 ; Setup the stack pointer to point to address 0x0070
         out spl,R16
         ldi R16,0x00
         out sph,R16

         ldi R16,0x01     ;make PB0 output
         out DDRB,R16

         ldi R16,0x00    ; make PORTD as input
         out DDRD, R16

         ldi R16,0x08    ; use pull up resistor for PD3
         out PORTD,R16

         in R16,MCUCR
         ori R16,0x08     ; set INT1 to falling edge sensitive
         out MCUCR,R16    ; use OR so that other bits are not affected

         in R16,GICR
         ori R16, 0x80    ; enable INT1 interrupt
         out GICR,R16

         ldi R16,0x00     ; Turn off LED
         out PORTB,R16

         sei               ; enable interrupts

         indefiniteloop: rjmp indefiniteloop

         INT1_ISR:             ; INT1 Interrupt handler or ISR
```

```
                in R16,SREG    ; save status register
                        push R16

                        ldi R16,0x0a   ; blink LED 10 times
                        mov R0,R16

        c1: ldi R16,0x01    ;Turn ON LED
            out PORTB,R16

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

                 ldi R16,0x00   ; Turn OFF LED
                 out PORTB,R16

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

                 DEC R0
                 BRNE c1    ; check if LED has blinked 10 times

                 pop  R16      ; retrive status register
                 out SREG, R16

                 RETI          ; go back to main program
```

Listing 3: Code for INT1 in ASM

4.2 Code in C Language

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

ISR (INT1_vect)
{
        // Write your ISR here to blink the LED 10 times
        // with ON and OFF interval of 1 second each
        for(int i=0; i<10; i=i+1)
        {
                //PB0 is set to 1 for 1 sec.
```

```c
                PORTB = 0x01;
                _delay_ms(1000);
                //PB0 is set to 0 for 1 sec.
                PORTB = 0x00;
                _delay_ms(1000);
        }
}

int main (void)
{
        //port i/o declarations
        DDRD = 0x00;
        DDRB = 0x01;
        MCUCR = 0x02;
        GICR = 0x80;
        PORTB = 0x00;

        //set interrupt flag of SREG
        sei();

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

Listing 4: Code for INT1 in C

4.3 Outputs

Video for INT1 ASM:
https://drive.google.com/file/d/1GkrKFJhulHYAP-31yp1crPQSms0TnDoy/view?usp=drivesdk

Video for INT1 C:
https://drive.google.com/file/d/1GmHhMud-DTHZ62_u5e-77DaNd0poouPf/view?usp=drivesdk

4.4 Explanation

- The ATmega 8 IC's pin 5 houses interrupt number 1, and the rest of the circuit is connected similarly.
- INT1's interrupt vector address. Jump to the ISR or place your ISR here.
- Set the stack pointer to point at address 0x0070, set PB0 to be an output, PORTD to be an input, and PD3 to be pulled up using a resistor.
- To ensure that other bits are not impacted, set INT1 to falling edge sensitive use OR.
- Save the status register and blink the LED ten times in the INT1 Interrupt handler or ISR. Interrupt.

5 Learning Outcomes

By doing this experiment we were able to:

- Learn how to program in assembly language and how to burn code into board.
- Get familiar with the software Atmel Studio and AVR Burn-OMAT.
- By using interrupts 0 and 1 with a 50% duty cycle, we learned how to make an LED blink using ASM and the C language.