EE2016: Experiment 6

Group:19

Sparsh Gupta	Charan Srikkanth	Kaushik Iyer
EE23B117	EE23B127	EE23B135

September 11, 2024

1 Introduction

This experiment focuses on how interrupts work.The main aim is to run a seperate piece of code when a button or key is pressed.

2 Objectives

The primary objectives of this experiment are:

1. Implement Interrupts in C code using ATMEGA 8 by transferring control from a white LED glowing to making an red LED blink when a push button is pressed.
2. Implement a similar interrupt function in Assembly language with the same ATMEGA 8.

3 Interrupt Handling in C code

The main way we do this is:

1. Set the frequency as 8Mhz.Import all required libraries.
2. Connect the white LED to PB0 and the Red LED to PB1.Set DDRB to 0x03 to allow output to both white and red LED.
3. Set the GICR 6th bit to 1 (which is INT0) so as to allow interrupts from INT0.The 0th(IVCE) and 1st bit(IVSEL) are set to 0 to enable interrupts.
4. Connect the push button to PD2(INT0) and set DDRD to 0 to allow input from all D ports.
5. Set SREG to 0x80 to enable the global interrupt flag.
6. When an interrupts happens the ISR function is called inside this function we call cli() which clears the global interrupt flag which disables interrupts when the ISR function is run.
7. At the End of the ISR function the sei() function is called which enables the global interrupt flag.This again enables interrupts.

3.1 C code

```
#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();
}
```

4 Interrupt Handling in Assembly

1. The logic is similar to the C code.However there are some additional things we have to do.
2. We set up the stack pointer so as to store the state at which we left the main code in.For example let us say some flags were set in the main code and then the interrupt was called.During the interrupt code some new flags were called and the old flags were removed.This will cause errors when we jump back to the main code.So in order to prevent this we store the state of SREG in the stack(initially when the interrupt is called) and when we are at the end of interrupt code we update SREG to its value in the stack.
3. In order to have a delay we run two nested loops (0xFF*0xFF) times which gives a delay of (0xFF*0xFF) clock cycles.

4.1 Assembly Code

```
.ORG 0x00
RJMP main ; on reset, program starts here

.ORG 0x02 ; Interrupt vector address for INT1.
RJMP int1_ISR ;

main:
;; Set the stack pointer to point to address 0x0070
LDI R16, 0x70
OUT spl, R16
LDI R16, 0X00
OUT sph, R16

;; DDRB = 3; // Set B0 and B1 as outputs
LDI R16, 0x03
OUT DDRB,R16

;; DDRD = 0; // Set PORTD to be input
LDI R16, 0x00
OUT DDRD,R16

;; Turn off LED
LDI R16, 0X00
OUT PORTB, R16

;; Enable INT1 Interrupt
LDI R16, 0x80
OUT GICR, R16

;; Enable interrupts
SEI

indefiniteloop: ;; Loop that runs forever
;; Make LED1 glow
LDI R16, 0X01;
OUT PORTB, R16

RJMP indefiniteloop

;; INT1 interrupt handler or ISR
int1_ISR:
CLI ;; Disable interrupts

;; Save SREG to the stack
IN R16, SREG
PUSH R16

LDI R21, 10 ; Counter
;; Do the following loop 10 times
blink:
;; Turn the LED on
LDI R16, 0x02
OUT PORTB, R16
```

```
;; Some delay (0xFF * 0xFF clock cycles)
LDI R16, 0xFF ; Counter
;; Do the following loop 0xFF times
delay_loop_outer_1:
LDI R17, 0xFF

;; Do the following loop 0xFF times
delay_loop_inner_1:
DEC R17
BRNE delay_loop_inner_1

DEC R16
BRNE delay_loop_outer_1

;; Turn off the LED
LDI R16, 0X00
OUT PORTB, R16

;; Some delay (0xFF * 0xFF clock cycles)
LDI R16, 0xFF ; Counter
;; Do the following loop 0xFF times
delay_loop_outer_2:
LDI R17, 0xFF

;; Do the following loop 0xFF times
delay_loop_inner_2:
DEC R17
BRNE delay_loop_inner_2

DEC R16
BRNE delay_loop_outer_2

DEC R21
BRNE blink

;; Reset SREG ()
POP R16; retrieve status register
OUT SREG, R16

SEI ;; Enable interrupts
RJMP indefiniteloop ;; Go back to the loop
```

5 Comments

1. The first Issue we faced was with wiring we hadnt connected the right ports. This has caused some issues
2. We forgot about using a pull up resistor for the pushbutton as input so the pin was floating which was giving wierd outputs.
3. We were slightly confused on how to implement the Assembly program.However with some searching we were able to get a grasp on how interrupts work and learnt about how to use the stack.