EE2016 : Microprocessor Theory and Lab

Lab Experiment # 4

INTERRUPTS IN ATMEL AVR ATMEGA THROUGH ASSEMBLY PROGRAMMING

Batch 35

Sarathchandra Koppera EE21B073

Vikas Gollapalli EE21B051

Aim

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

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

(ii) 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).

(iii) 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. Also, one needs to implement all of the above using C-interface.

Equipment Required

Since this is an emulation-based experiment, we need only a PC with the following software: Atmel studio simulation software.

The list of equipment, software, components required are:

1. Atmel Atmega8 Microcontroller chip

2. Bread board with hardware components

3. A PC with Microchip Studio simulation software loaded

4. Data / power cables

Code for INT 0 with ASM

```
#include "m8def.inc"
.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, 0x02; //
 OUT MCUCR, R16;
 IN R16, GICR; //Load GICR register
 ORI R16, 0x40;
 OUT GICR, R16;
 LDI R16, 0x00;
 OUT PORTB, R16;
 SEI
ind_loop:
 RJMP ind_loop;
int0_ISR:
 IN R16, SREG;
 PUSH R16;
 LDI R16, 0x0A;
 MOV R0, R16;
 c1: LDI R16, 0x01; //
 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
```

Code for INT 0 with C

```
//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)
 {
 //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.
 }
}
```

Code for INT 1 with ASM

```
//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 R16,0x02 ;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
```

Code for INT 1 with C

```
#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.
 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.
 }
}
```

Explanation

For INT 0

• The INT0 interrupt detects different levels and level changes on the INT0 input, which refers to pin 4 in an ATmega8

. • If the respective interrupt is enabled ,the interrupt branches to the INT0 vector. The bits ISC01 and ISC00 permits the selection of states or changes which trigger this interrupt.

• The primary objective is to detect if the counter is at zero. If it does, then the T flag is checked. If set, a new cycle starts and the counter is set to six.

• If the cycle counter is not zero, the bTo flag is checked (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.

For INT 1

• Interrupt 1 is in pin5 of ATmega 8 IC and the rest of the circuit is similar in connection.

• Interrupt vector address for INT1. Put your ISR here or jump to the ISR.

• Setup the stack pointer to point to address 0x0070, make PB0 output, make PORTD as input, use pull up resistor for PD3.

• Set INT1 to falling edge sensitive use OR in order to ensure that other bits are not affected enable INT1.

• INT1 Interrupt handler or ISR, save status register and blink LED 10 times. Interrupt.

Learning Outcomes

- Learning how to burn code into the hardware
- Getting used to code in assembly language and then use Burn o mat to implement the code
- By using int 0 and 1 with 50% duty cycle we learnt to blink the LED using ASM and C language

Outputs

Please refer to these videos for the outputs of the experiment

https://drive.google.com/drive/folders/1vQTKNX8ZBGJwZ5Rve1ldPHi1AAzE0LcQ?usp=sharing