

EE2016 : Microprocessor Theory and Lab

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

Mandla Siva Manoj, EE21B083  
Ajo George, EE21B009  
Group 3

October 15, 2022

## Contents

<b>1</b>	<b>Aim of the Experiment</b>	<b>1</b>
<b>2</b>	<b>Problems</b>	<b>1</b>
<b>3</b>	<b>INT 0</b>	<b>2</b>
3.1	Code in Assembly Language . . . . .	2
3.2	Code in C Language . . . . .	3
3.3	Outputs . . . . .	4
3.4	Explanation . . . . .	4
<b>4</b>	<b>INT 1</b>	<b>6</b>
4.1	Code in Assembly Language . . . . .	6
4.2	Code in C Language . . . . .	7
4.3	Outputs . . . . .	9
4.4	Explanation . . . . .	9
<b>5</b>	<b>Learning Outcomes</b>	<b>10</b>

## 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

```
1 //ASM program to implement external input INTO in AVR
2 //switch to PD2
3 //LED to PB0
4
5 .nolist
6 .include "m8def.inc"
7 .list
8
9 .org 0
10 rjmp reset      ; on reset, program starts here
11
12 .org 0x0002      ; Interrupt vector address for INTO. Put your ISR here or
   ↪ jump
13 rjmp INTO_ISR   ; to the ISR
14
15 .org 0x0100
16
17 reset:
18     ldi R16,0x70 ; Setup the stack pointer to point to address 0x0070
19     out spl,R16
20     ldi R16,0x00
21     out sph,R16
22
23     ldi R16,0x01 ;make PB0 output
24     out DDRB,R16
25
26     ldi R16,0x00 ; make PORTD as input
27     out DDRD, R16
28
29     ldi R16,0x04 ; use pull up resistor for PD2
30     out PORTD,R16
31
32     in R16,MCUCR
33     ori R16,0x02 ; set INTO to falling edge sensitive
34     out MCUCR,R16 ; use OR so that other bits are not affected
35
36     in R16,GICR
37     ori R16, 0x40 ; enable INTO interrupt
38     out GICR,R16
39
40     ldi R16,0x00 ; Turn off LED
41     out PORTB,R16
42
43     sei          ; enable interrupts
44
45     indefiniteloop: rjmp indefiniteloop
46
```

```

47      INTO_ISR:                ; INTO Interrupt handler or ISR
48          in R16,SREG          ; save status register
49          push R16
50
51          ldi R16,0x0a         ; blink LED 10 times
52          mov R0,R16
53
54      c1: ldi R16,0x01         ; Turn ON LED
55          out PORTB,R16
56
57          LDI R16,0xFF         ; delay
58      a1: LDI R17,0xFF
59      a2: DEC R17
60          BRNE a2
61          DEC R16
62          BRNE a1
63
64          ldi R16,0x00         ; Turn OFF LED
65          out PORTB,R16
66
67          LDI R16,0xFF         ; delay
68      b1: LDI R17,0xFF
69      b2: DEC R17
70          BRNE b2
71          DEC R16
72          BRNE b1
73
74          DEC R0
75          BRNE c1             ; check if LED has blinked 10 times
76
77          pop R16              ; retrieve status register
78          out SREG, R16
79
80          RETI                 ; go back to main program

```

Listing 1: Code for INT0 in ASM

### 3.2 Code in C Language

```

1  //C program for INT0
2  #define F_CPU 1000000
3  #include <avr/io.h>
4  #include <util/delay.h>
5  #include <avr/interrupt.h>
6
7  ISR (INT0_vect)
8  {
9      // Write your ISR here to blink the LED 10 times
10     // with ON and OFF interval of 1 second each
11     for(int i=0; i<10; i=i+1)

```

```

12     {
13         //PB0 is set to 1 for 1 sec.
14         PORTB = 0x01;
15         _delay_ms(1000);
16         //PB0 is set to 0 for 1 sec.
17         PORTB = 0x00;
18         _delay_ms(1000);
19     }
20 }
21
22 int main (void)
23 {
24     //port i/o declarations
25     DDRD = 0x00;
26     DDRB = 0x01;
27     MCUCR = 0x02; //check
28     GICR = 0x40;
29     PORTB = 0x00;
30
31     //set interrupt flag of SREG
32     sei();
33
34     while (1)
35     {
36         //To keep the program running forever.
37     }
38 }

```

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

```
1 //ASM program to implement external input INT1 in AVR
2 //switch to PD3
3 //LED to PB0
4
5 .nolist
6 .include "m8def.inc"
7 .list
8
9 .org 0
10 rjmp reset ; on reset, program starts here
11 .org 0x0002 ; Interrupt vector address for INT1. Put your ISR here or
   ↪ jump
12 rjmp INT1_ISR ; to the ISR
13
14 .org 0x0100
15
16 reset:
17     ldi R16,0x70 ; Setup the stack pointer to point to address 0x0070
18     out spl,R16
19     ldi R16,0x00
20     out sph,R16
21
22     ldi R16,0x01 ;make PBO output
23     out DDRB,R16
24
25     ldi R16,0x00 ; make PORTD as input
26     out DDRD, R16
27
28     ldi R16,0x08 ; use pull up resistor for PD3
29     out PORTD,R16
30
31     in R16,MCUCR
32     ori R16,0x08 ; set INT1 to falling edge sensitive
33     out MCUCR,R16 ; use OR so that other bits are not affected
34
35     in R16,GICR
36     ori R16, 0x80 ; enable INT1 interrupt
37     out GICR,R16
38
39     ldi R16,0x00 ; Turn off LED
40     out PORTB,R16
41
42     sei ; enable interrupts
43
44     indefiniteloop: rjmp indefiniteloop
45
46     INT1_ISR: ; INT1 Interrupt handler or ISR
```



```

47         in R16,SREG    ; save status register
48         push R16
49
50         ldi R16,0x0a    ; blink LED 10 times
51         mov R0,R16
52
53     c1: ldi R16,0x01     ;Turn ON LED
54         out PORTB,R16
55
56         LDI R16,0xFF    ;delay
57     a1: LDI R17,0xFF
58     a2: DEC R17
59         BRNE a2
60         DEC R16
61         BRNE a1
62
63         ldi R16,0x00    ; Turn OFF LED
64         out PORTB,R16
65
66         LDI R16,0xFF    ;delay
67     b1: LDI R17,0xFF
68     b2: DEC R17
69         BRNE b2
70         DEC R16
71         BRNE b1
72
73         DEC R0
74         BRNE c1    ; check if LED has blinked 10 times
75
76         pop R16        ; retrieve status register
77         out SREG, R16
78
79         RETI            ; go back to main program

```

Listing 3: Code for INT1 in ASM

## 4.2 Code in C Language

```

1  #define F_CPU 1000000
2  #include <avr/io.h>
3  #include <util/delay.h>
4  #include <avr/interrupt.h>
5
6  ISR (INT1_vect)
7  {
8      // Write your ISR here to blink the LED 10 times
9      // with ON and OFF interval of 1 second each
10     for(int i=0; i<10; i=i+1)
11     {
12         //PB0 is set to 1 for 1 sec.

```

```

13         PORTB = 0x01;
14         _delay_ms(1000);
15         //PB0 is set to 0 for 1 sec.
16         PORTB = 0x00;
17         _delay_ms(1000);
18     }
19 }
20
21 int main (void)
22 {
23     //port i/o declarations
24     DDRD = 0x00;
25     DDRB = 0x01;
26     MCUCR = 0x02;
27     GICR = 0x80;
28     PORTB = 0x00;
29
30     //set interrupt flag of SREG
31     sei();
32
33     while (1)
34     {
35         //To keep the program running forever.
36     }
37 }

```

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](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.