# EE2016: Microprocessors Lab Experiment # 4: Interrupts in Atmel AVR through Assembly Programming

Amizhthni PRK, EE21B015 Nachiket Dighe, EE21B093

October 15, 2022

# Contents

1	Assembly 1		
	1.1	Aim:	1
	1.2	Procedure	1
	1.3	Circuit diagram	1
	1.4	Code	2
		1.4.1 Using Interrupt int0	2
		1.4.2 Using Interrupt int1	4
	1.5	Input and Output	6
2	$\mathbf{C}$ $\mathbf{c}$	ode	6
	2.1	Aim:	6
	2.2	Procedure	7
	2.3	Code	7
		2.3.1 Using Interrupt int0	7
		2.3.2 Using Interrupt int1	8
3	Exp	planation	E
4	Vid	eos and Outputs	9
$\mathbf{L}_{i}$	ist o	of Figures	
	1	Output circuit board	]
	2	Output of LED blinking	6

# 1 Assembly

#### 1.1 Aim:

Using Atmel AVR assembly language programming:

- 1. Generate an external (logical) hardware interrupt using an emulation of a push button switch.
- 2. 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).

#### 1.2 Procedure

- Set the corresponding pins as input and output.
- Wire the circuit as shown in figure.
- Write code in AVR aseembly language to emulate the push button so that the LED blinks for 10secs.

#### 1.3 Circuit diagram

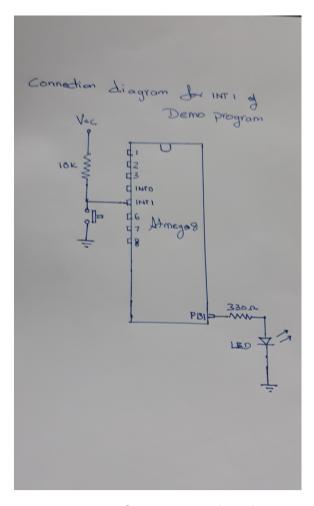


Figure 1: Output circuit board

#### 1.4 Code

#### 1.4.1 Using Interrupt int0

The AVR assembly code for controlling the LED and making it Blink for 10secs using **int0** Interrupt is shown below in linking 1:

```
.org 0x0000
  rjmp reset
3
   .org 0x0001
4
  rjmp int0_ISR
5
   .org 0x0100
  reset:
9
            ;Loading stack pointer address
10
11
            LDI R16,0x70
12
            OUT SPL,R16
13
            LDI R16,0x00
            OUT SPH, R16
15
16
            ;Interface port B pin0 to be output
17
            ;so to view LED blinking
18
            LDI R16,0x01
19
            OUT DDRB,R16
20
21
           LDI R16,0x00
22
            OUT DDRD, R16
23
24
            ;Set MCUCR register to enable low level interrupt
25
           LDI R16, 0x00
26
            OUT MCUCR, R16
27
28
            ;Set GICR register to enable interrupt 1
29
            LDI R16, 1<<INTO
30
            OUT GICR, R16
32
           LDI R16,0x00
33
            OUT PORTB, R16
34
35
            SEI
36
37
   ind_loop:
           rjmp ind_loop
38
39
40
41
42
   ;Interupt Service Routine
44
45
```

```
int0_ISR:
46
            IN R16, SREG
47
            PUSH R16
48
49
            LDI R16,0x0A ;For blinking LED 10 times
            MOV RO,R16
51
52
               LDI R16, 0x01; Setting LED to High (ON)
   c1:
53
            OUT PORTB, R16
54
55
   ;Program To delay
56
   ;start
57
            LDI R16, 200
58
   a1:
                LDI R17, 100
59
                LDI R18, 10
   a2:
60
               NOP
   a3:
                                     ;to waste cycle
61
            NOP
63
            DEC R18
64
            BRNE a3
65
66
            DEC R17
67
            BRNE a2
68
69
            DEC R16
70
            BRNE a1
71
72
            LDI R16, 0x00 ;Setting LED to Low (Off)
73
            OUT PORTB, R16
   ;end
75
   ;Program To delay
76
   ;start
77
            LDI R16, 200
78
                LDI R17, 100
   b1:
                LDI R18, 10
   b2:
80
                NOP
   b3:
                                      ;to waste cycle
81
            NOP
82
83
            DEC R18
84
            BRNE b3
            DEC R17
87
            BRNE b2
88
89
            DEC R16
90
            BRNE b1
91
   ;end
92
93
            DEC RO
94
            BRNE c1 ;To repeat blinking 10 times
```

```
96
97
POP R16
98
OUT SREG, R16 ;restoring values before interruption.
99
100
RETI
```

Listing 1: Using int0 Interrupt in AVR

#### 1.4.2 Using Interrupt int1

The AVR assembly code for controlling the LED and making it Blink for 10secs using **int1** Interrupt is shown below in linking 2:

```
.org 0x0000
  rjmp reset
   .org 0x0002
  rjmp int1_ISR
   .org 0x0100
   reset:
9
            ;Loading stack pointer address
10
11
            LDI R16,0x70
            OUT SPL,R16
            LDI R16,0x00
14
            OUT SPH, R16
15
16
            ;Interface port B pin0 to be output
17
            ;so to view LED blinking
            LDI R16,0x01
19
            OUT DDRB,R16
20
21
            LDI R16,0x00
22
            OUT DDRD, R16
23
            ;Set MCUCR register to enable low level interrupt
25
            LDI R16, 0x00
26
            OUT MCUCR, R16
27
28
            ;
Set GICR register to enable interrupt \boldsymbol{1}
            LDI R16, 1<<INT1
30
            OUT GICR, R16
31
32
            LDI R16,0x00
33
            OUT PORTB, R16
34
           SEI
37 ind_loop:
```

```
rjmp ind_loop
38
39
40
41
43
   ;Interupt Service Routine
44
45
   int1_ISR:
46
            IN R16, SREG
47
            PUSH R16
48
49
            LDI R16,0x0A ;For blinking LED 10 times
50
            MOV RO,R16
51
52
               LDI R16, 0x01; Setting LED to High (ON)
   c1:
53
            OUT PORTB, R16
55
   ;Program To delay
56
   ;start
57
            LDI R16, 200
58
   a1:
                LDI R17, 100
59
   a2:
                LDI R18, 10
   a3:
               NOP
                                     ;to waste cycle
61
            NOP
62
63
            DEC R18
64
            BRNE a3
65
            DEC R17
67
            BRNE a2
68
69
            DEC R16
70
            BRNE a1
71
72
            LDI R16, 0x00 ;Setting LED to Low (Off)
73
            OUT PORTB, R16
74
75
   ;end
   ;Program To delay
76
   ;start
77
            LDI R16, 200
                LDI R17, 100
   b1:
79
   b2:
                LDI R18, 10
80
                NOP
   b3:
                                      ;to waste cycle
81
            NOP
82
            DEC R18
84
            BRNE b3
85
86
            DEC R17
```

```
BRNE b2
88
89
            DEC R16
90
            BRNE b1
91
   ;end
93
            DEC RO
94
            BRNE c1 ; To repeat blinking 10 times
95
96
            POP R16
97
            OUT SREG, R16 ; restoring values before interruption.
99
            RETI
100
```

Listing 2: Using int1 Interrupt in AVR

#### 1.5 Input and Output

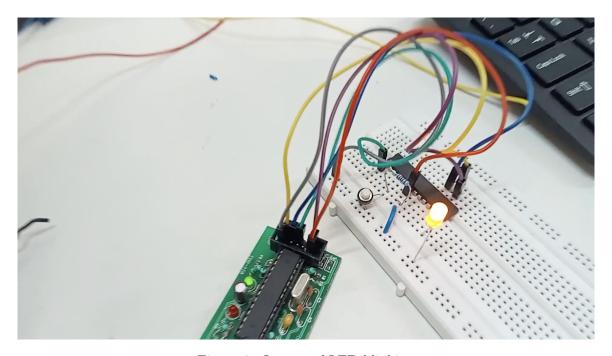


Figure 2: Output of LED blinking

# 2 C code

#### 2.1 Aim:

Using C language programming:

- 1. Generate an external (logical) hardware interrupt using an emulation of a push button switch.
- 2. 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).

#### 2.2 Procedure

- Set the corresponding pins as input and output.
- Wire the circuit as shown in figure.
- Write code in C language to emulate the push button so that the LED blinks for 10secs.

#### 2.3 Code

#### 2.3.1 Using Interrupt int0

The C language code for controlling the LED and making it Blink for 10secs using Interrupt int0 is shown in linking 3:

```
#define F_CPU 1000000
  #include <avr/io.h>
  #include <util/delay.h>
   #include <avr/interrupt.h>
  ISR (INT1_vect)
  {
7
           // ISR to blink the LED 10 times
           // ON and OFF interval of 1 second each
           int i;
10
           for(i=0; i<10; i++)
           {
                    //PBO is set to High for 1 sec.
13
                    PORTB = 0x01;
14
                    _delay_ms(1000);//Including delay of 1 sec
15
16
                    //PBO is set to Low for 1 sec.
                    PORTB = 0x00;
                    _delay_ms(1000);//Including delay of 1 sec
19
           }
20
   }
21
22
   int main (void)
23
   {
24
           DDRD = 0x00;
25
           DDRB = 0x01; //Make PBO as output
26
27
           MCUCR = 0x00; //Set MCUCR to level triggered
28
29
           GICR = (1<<INTO); //Enable interrupt 0
30
31
           PORTB = 0x00;
32
33
           sei(); // global interrupt flag
34
           while (1)
           {
36
                    //To keep the program running forever.
37
```

Listing 3: Using Interrupt int0 in C

#### 2.3.2 Using Interrupt int1

The C language code for controlling the LED and making it Blink for 10secs using Interrupt int1 is shown in linking 4:

```
#define F_CPU 1000000
   #include <aur/io.h>
   #include <util/delay.h>
   #include <avr/interrupt.h>
  ISR (INT1_vect)
6
  {
7
           // ISR to blink the LED 10 times
           // ON and OFF interval of 1 second each
           int i;
10
           for(i=0; i<10; i++)
11
           {
12
                    //PBO is set to High for 1 sec.
13
                    PORTB = 0x01;
14
                    _delay_ms(1000);//Including delay of 1 sec
15
                    //PBO is set to Low for 1 sec.
17
                    PORTB = 0x00;
18
                    _delay_ms(1000);//Including delay of 1 sec
19
           }
20
  }
21
22
   int main (void)
23
   {
24
           DDRD = 0x00;
25
           DDRB = 0x01; //Make PBO as output
26
           MCUCR = 0x00; //Set MCUCR to level triggered
28
29
           GICR = (1<<INT1); //Enable interrupt 1
30
31
           PORTB = 0x00;
32
33
           sei(); // global interrupt flag
34
           while (1)
35
36
                    //To keep the program running forever.
37
           }
38
  }
39
```

Listing 4: Using Interrupt int1 in C

### 3 Explanation

- 1. The external interrupts are triggered by the INT0, and INT1 pins. Observe that, if enabled, the interrupts will trigger even if the INT0 or 1 pins are configured as outputs.
- 2. This feature provides a way of generating a software interrupt. The external interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU Control Register MCUCR.
- 3. When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 and INT1 requires the presence of an I/O clock.
- 4. Low level interrupts on INT0/INT1 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

## 4 Videos and Outputs

The Output Video of the LED blinking using Interrupts is given in the Video Link