# Experiment 2: Interrupts and Timers in Atmel AVR Atmega

**Aim**: Using Atmel AVR assembly language programming, implement interrupts and timers in Atmel Atmega microprocessor. The main constraint is that, it should be emulation only (due to ongoing pandemic). 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.

## Questions to be Done:

You are given two files (apart from this handout in the directory EE2016F21Exprmnt2AVR.IntrrptCIntrfcng in moodle): EE2016F21Exp2int1.asm and EE2016F21Exp2.int1.c The former one is an assembly program which implements interrupt using int1, while the later one is a 'C' program using int1.c

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

# Logic Execution:

Main task of the program is to blink LED which is connected to output ports 10 times with one second delay between ON and OFF.

We have to do this task in AVR assembly language and in C language. Here the concept of Interrupts has been involed. In I/O circuit we have two options available for giving interrupts- INT0 and INT1. We will discuss in both ways.

**A) FOR INT0:**

Steps:-

1) Switch is connected to PIN3 of PORTD which is used to take inputs from that pin(To get external interrupt response from switch)
2) To make PD2 as input pin we need to LOAD 0x00 value in DDRD
3) LED is connected to pin0 of portB to get output response from PB0.
4) To make PB0 as output pin we need to LOAD 0x01 value in DDRB
5) As the interrupt is LOW LEVEL TRIGGERD we also need to enable ISC00 and ISC01 bits in MCUCR register.(This can be done loading 0x00 in MCUCR register).
6) Before executing ISR we need to enable External hardware Interrupts, this can done using GICR register in which INT0 bit is made to set.(For this we load 0x040 in GICR register to enable INT1)
7) Global interrupt enable flag(I) of status register(SREG) is set. If there is no interrupt then main program will be continued
8) As the interrupt is enabled the current instruction in Main program gets finished.
9) Present address of program counter in main program is get pushed into stack pointer address. All the contents of the main program that are to be executed next will be stored in stack pointers by decrementing the stack pointer address for each instruction.(SPL and SPH are 16 bit stack pointers used to store address of PC).
10) Upon calling of ISR the Global interrupt flag (I) is retained to “0”, So that another Interrupt may not occur.
11) Now we have to program ISR_INT0 in such a way that, it should ON and OFF the LED which is connected to PB0 with delay of one second for Ten times.

i) First LOAD PORTB with the value 0x01, such that pin0 will get active high logic(LED ON).
ii) Now we write instructions to get the delay of one second using registers as counters and looping them.
iii) After one second delay we LOAD PORTB with the value 0x00, such that pin0 will get active low logic( LED OFF)
iv) Again write the same instructions to make the delay of one second which were used in step-ii.
v) The above four steps make the LED to ON and OFF for one time with one second delay in each operation.

vi) To make LED blink for ten times we use a register as a counter(let us take R0=10). Using this R0 as counter we can repeat the above five steps for ten times which makes the LED to blink for 10 times. This step completes the ISR_INT0 sub routine.
12) After completion of ISR we POP out the contents of SREG registers which were stacked earlier before interrupt and continue the main program
13) Also we set the Interrupt flag using RETI instruction at the end of the program to allow another interrupt to occur.

**B) FOR INT1:**

Steps:
1) Switch is connected to PIN3 of PORTD which is used to take inputs from that pin(To get external interrupt response from switch)
2) To make PD3 as input pin we need to LOAD 0x00 value in DDRD
3) LED is connected to pin0 of portB to get output response from PB0.
4) To make PB0 as output pin we need to LOAD 0x01 value in DDRB
5) As the interrupt is LOW LEVEL TRIGGERD we also need to enable ISC10 and ISC11 bits in MCUCR register.(This can be done loading 0x00 in MCUCR register).
6) Before executing ISR we need to enable External hardware Interrupts, this can done using GICR register in which INT1 bit is made to set.(For this we load 0x080 in GICR register to enable INT1)
7) Global interrupt enable flag(I) of status register(SREG) is set. If there is no interrupt then main program will be continued
8) As the interrupt is enabled the current instruction in Main program gets finished.
9) Present address of program counter in main program is get pushed into stack pointer address. All the contents of the main program that are to be executed next will be stored in stack pointers by decrementing the stack pointer address for each instruction.(SPL and SPH are 16 bit stack pointers used to store address of PC).
10) Upon calling of ISR the Global interrupt flag (I) is retained to “0”, So that another Interrupt may not occur.
11) Now we have to program ISR_INT1 in such a way that, it should ON and OFF the LED which is connected to PB0 with delay of one second for Ten times.
i) First LOAD PORTB with the value 0x01, such that pin0 will get active high logic(LED ON).
ii) Now we write instructions to get the delay of one second using registers as counters and looping them.
iii) After one second delay we LOAD PORTB with the value 0x00, such that pin0 will get active low logic( LED OFF)
iv) Again write the same instructions to make the delay of one second which were used in step-ii.
v) The above four steps make the LED to ON and OFF for one time with one second delay in each operation.
vi) To make LED blink for ten times we use a register as a counter(let us take R0=10). Using this R0 as counter we can repeat the above five steps for ten times which makes the LED to blink for 10 times. This step completes the ISR_INT1 sub routine.
12) After completion of ISR we POP out the contents of SREG registers which were stacked earlier before interrupt and continue the main program
13) Also we set the Interrupt flag using RETI instruction at the end of the program to allow another interrupt to occur.

**This Logic can be used to write program in both assembly language and C language.

## Assembly Program for INT0:

```
; INT0.asm
;
; Created: 22-09-2021
; Author : YASHWANTH
;


.org 0x0000
rjmp reset

.org 0x0002 ;Address for INT0 port.
rjmp int0_ISR ;jumping to another memory location due to only one address is available after 0x0002

.org 0x0100  ;main program starts from memory location 0x100

reset:
         ;Loading stack pointer address
      LDI R16,0x70
         OUT SPL,R16
         LDI R16,0x00
         OUT SPH,R16

         ;Interface port B pin0 to be output
         ;so to view LED blinking

         LDI R16,0x00  ;clearing R16 register
         OUT DDRD,R16  ;Making portB as input port
         SBI PORTD,2   ;Activating pull up for pin2 in PortD
      LDI R16,0x01  ;loading register with hex 1
         OUT DDRB,R16  ;Pin0 of PORTB as output port

         ;Set MCUCR register to enable low level interrupt
         IN R16, MCUCR ;Loading contents of MCUCR register to R16
         LDI R16,0x00
         OUT MCUCR,R16 ; ISC00,ISC01 contents are made to 0,0

         ;Set GICR register to enable interrupt 0
         IN R16, GICR
         LDI R16,0x40         ;Loading 0x40 to R16
         OUT GICR,R16         ;Changing MCUCR to GICR and making INT0 of GICR set.

         LDI R16,0x00
         OUT PORTB,R16

         SEI

ind_loop:rjmp ind_loop

int0_ISR:IN R16,SREG
              PUSH R16

              LDI R16,0x0A   ;loading (1010)2 in R16
              MOV R0,R16     ; copy R16 to R0


              ; Below loops are to make LED blink for 1 sec
       T1:    LDI R16,0x01
```

```
            OUT PORTB,R16  ; Making pin0 of PORTB as active high(lets say ON condition)



;------------------------ DELAY---------------------

            // 4*25*50*200 = 1000000(1s delay)       (0.03% error)

            LDI R20,200
      L1:   LDI R21,100
      L2:   LDI R22,25
      L3:
          NOP
            DEC R22
            BRNE L3

            DEC R21
            BRNE L2

            DEC R20
            BRNE L1
   ;-------------------------------------------------------

            LDI R16,0x00
            OUT PORTB,R16 ; pin0 of PORTB made as active low(lets Say OFF)

      ;------------------------ DELAY---------------------

      // ;4*25*50*200 = 1000000(1s delay)


            LDI R20,200      ; R20,R21,R22 acts as counters to bring delay of 1s. Numbers are
allotted in such a way to bring 10^6 microseconds delay
      M1:   LDI R21,50
      M2:   LDI R22,25
      M3:
            NOP              ; This loop makes a delay of 4 microseconds
            DEC R22
            BRNE M1

            DEC R21
            BRNE M2

            DEC R20
            BRNE M3

           ;------------------------ DELAY---------------------

            DEC R0
          BRNE T1      ; Above loops are repated upto 10 times, till R0 becomes 0
            POP R16      ; After completion of ISR, we POP out to main program
            OUT SREG, R16 ; POP SREG register to continue the main program which has been stopped

            RETI
```

## Assembly Program For INT1:

```
;
; INT1.asm
;
; Created: 22-09-2021 16:46:39
; Author : YASH
;


..org 0x0000
rjmp reset

.org 0x0004 ;Address for INT0 port in IVT
rjmp int0_ISR ;jumping to another memory location due to only one address is available after 0x0002

.org 0x0100  ;main program starts from memory location 0x100

reset:
         ;Loading stack pointer address
      LDI R16,0x70
         OUT SPL,R16
         LDI R16,0x00
         OUT SPH,R16

         ;Interface port B pin0 to be output
         ;so to view LED blinking

         LDI R16,0x00  ;clearing R16 register
         OUT DDRD,R16  ;Making portB as input port
         SBI PORTD,3   ;Activating pull up for pin3 in PortD
      LDI R16,0x01  ;loading register with hex 1
         OUT DDRB,R16  ;Pin0 of PORTB as output port

         ;Set MCUCR register to enable low level interrupt
         IN R16, MCUCR ;Loading contents of MCUCR register to R16
         LDI R16,0x00
         OUT MCUCR,R16 ; ISC00,ISC01 contents are made to 0,0

         ;Set GICR register to enable interrupt 0
         IN R16, GICR
         LDI R16,0x80         ;R16--> $80
         OUT GICR,R16         ;enabling INT1 bit of GICR

         LDI R16,0x00
         OUT PORTB,R16

         SEI

ind_loop:rjmp ind_loop

int0_ISR:IN R16,SREG
               PUSH R16

               LDI R16,0x0A   ;loading (1010)2 in R16
               MOV R0,R16     ; copy R16 to R0
```

```
              ; Below loops are to make LED blink for 1 sec
       T1:    LDI R16,0x01
              OUT PORTB,R16  ; Making pin0 of PORTB as active high(lets say ON condition)



;----------------------- DELAY--------------------


              // 4*25*50*200 = 1000000(1s delay)       (0.03% error)

              LDI R20,200
       L1:    LDI R21,100
       L2:    LDI R22,25
       L3:
            NOP
              DEC R22
              BRNE L3

              DEC R21
              BRNE L2

              DEC R20
              BRNE L1
    ;-------------------------------------------------------

              LDI R16,0x00
              OUT PORTB,R16 ; pin0 of PORTB made as active low(lets Say OFF)

       ;----------------------- DELAY--------------------

       // ;4*25*50*200 = 1000000(1s delay)


              LDI R20,200      ; R20,R21,R22 acts as counters to bring delay of 1s. Numbers are
allotted in such a way to bring 10^6 microseconds delay
       M1:    LDI R21,50
       M2:    LDI R22,25
       M3:
              NOP              ; This loop makes a delay of 4 microseconds
              DEC R22
              BRNE M1

              DEC R21
              BRNE M2

              DEC R20
              BRNE M3

             ;----------------------- DELAY--------------------

              DEC R0
            BRNE T1      ; Above loops are repated upto 10 times, till R0 becomes 0
              POP R16      ; After completion of ISR, we POP out to main program
              OUT SREG, R16 ; POP SREG register to continue the main program which has been stopped

              RETI
```

## INT0 in C language:

```
* INT0_C.c
 *
 * Created: 22-09-2021
 * Author : YASHWANTH
 */

#define F_CPU 1000000  // Clock frequency

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h> //Header files

ISR (INT0_vect)
{
	int i;
	for (i=1;i<=10;i++) // For blinking LED 10 times

	{
		PORTB|=(1<<0);  //  load 00000001 in portB such that pin0 gets active high
		_delay_ms(1000);   // delay of 1 sec
		PORTB&=~(1<<0);  // 00000000
		_delay_ms(1000);

	}


}
int main(void)
{
	//Set the input/output pins appropriately
	//To enable interrupt and port interfacing
	//For LED to blink
	DDRD&=~(1<<2);   //Set appropriate data direction for D
	PORTD|=(1<<2);   //Activating pull-up of PD2(int0)
	DDRB|=(1<<0);   //Make PB0 as output pin    (changed from $00 to $01)
	MCUCR&=~(1<<ISC11|1<<ISC10);  //Set MCUCR to level triggered which makes ISC00 and ISC01 to 0,0
	GICR|=(1<<6);   //Enabling Interrupt 0     (changed from $00 to $40)
	PORTB&=~(1<<0);
	sei();       //  Set Global Interrupt flag enable

	while (1) //wait
	{

	}
}
```

## INT1 in C language:

```
* INT1_C.c
 *
 * Created: 22-09-2021
 * Author : YASHWANTH
 */

#define F_CPU 1000000  // Clock frequency

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h> //Header files

ISR (INT0_vect)
{
	int i;
	for (i=1;i<=10;i++) // For blinking LED 10 times

	{
		PORTB|=(1<<0);  //  load 00000001 in portB such that pin0 gets active high
		_delay_ms(1000);   // delay of 1 sec
		PORTB&=~(1<<0);  // 00000000
		_delay_ms(1000);

	}


}
int main(void)
{
	//Set the input/output pins appropriately
	//To enable interrupt and port interfacing
	//For LED to blink
	DDRD&=~(1<<3);   //Making pin3 of portD active low
	PORTD|=(1<<3);   //Activating pull-up of PD3(int0)
	DDRB|=(1<<0);   //Make PB0 as output pin    (changed from $00 to $01)
	MCUCR&=~(1<<ISC01|1<<ISC00);  //Set MCUCR to level triggered which makes ISC10 and ISC11 to 0,0
	GICR|=(1<<7);   //Enabling Interrupt 1     (changed from $00 to $80)
	PORTB&=~(1<<0);
	sei();       //  Set Global Interrupt flag enable

	while (1) //wait
	{

	}
}
```

## Learning Outcomes:

I have learnt interrupt programming for blinking Led for 10 times using Assembly language program and C language program. In comparing both the languages C language tends to be more easier than assembly language programming. Also the memory usage in C programming is less compared to Assembly language and we have the availability of directives in C programming in which many functions (we have used delay function here) are stored as default.

## Conclusion:

After Switch ON, LED blinks with interval of one second Delay for 10 Times. Using of C language for programming is more convenient than Assembly language.

***Repot Submission By:***

***J. Yashwanth***

***EE20B048.***