EE2016
LAB 4 REPORT
B44

Ananya Das EE21B016
Papisetty Sailendra EE21B098

Aim

Using Atmel AVR assembly language programming, implement interrupts and DIP switches control in Atmel Atmega microprocessor.

The 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 equipment, software, and components required are

1. Atmel Atmega8 Microcontroller chip

2. Breadboard with hardware components

3. A PC with Microchip Studio simulation software loaded

4. Data/power cables Instructions related to Interrupts

Instructions related to Interrupts

Bit	7	6	5	4	3	2	1	0	
	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

General Interrupt Controller

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	–	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	–	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	–	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Table 18. Reset and Interrupt Vectors

Vector No.	Program Address[2]	Source	Interrupt Definition
1	0x000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, and Watchdog Reset
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
4	0x003	TIMER2 COMP	Timer/Counter2 Compare Match
5	0x004	TIMER2 OVF	Timer/Counter2 Overflow
6	0x005	TIMER1 CAPT	Timer/Counter1 Capture Event
7	0x006	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	0x007	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	0x008	TIMER1 OVF	Timer/Counter1 Overflow
10	0x009	TIMER0 OVF	Timer/Counter0 Overflow
11	0x00A	SPI, STC	Serial Transfer Complete
12	0x00B	USART, RXC	USART, Rx Complete
13	0x00C	USART, UDRE	USART Data Register Empty
14	0x00D	USART, TXC	USART, Tx Complete
15	0x00E	ADC	ADC Conversion Complete
16	0x00F	EE_RDY	EEPROM Ready
17	0x010	ANA_COMP	Analog Comparator
18	0x011	TWI	Two-wire Serial Interface
19	0x012	SPM_RDY	Store Program Memory Ready

BOOTRST[1]	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x000	0x001
1	1	0x000	Boot Reset Address + 0x001
0	0	Boot Reset Address	0x001
0	1	Boot Reset Address	Boot Reset Address + 0x001

Connections

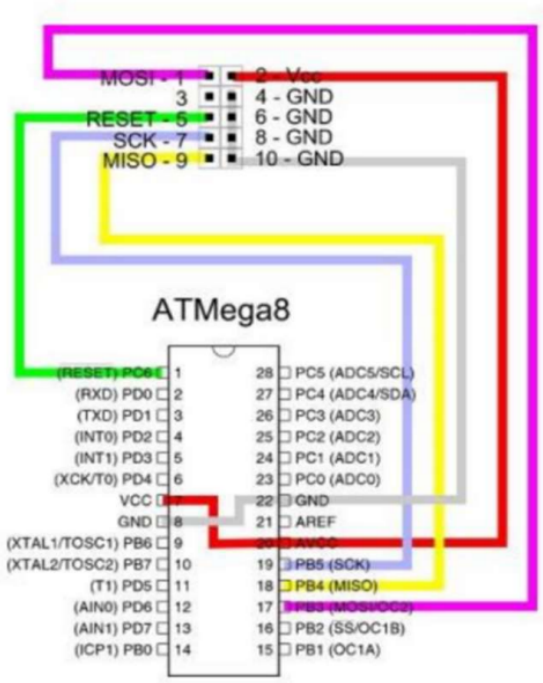
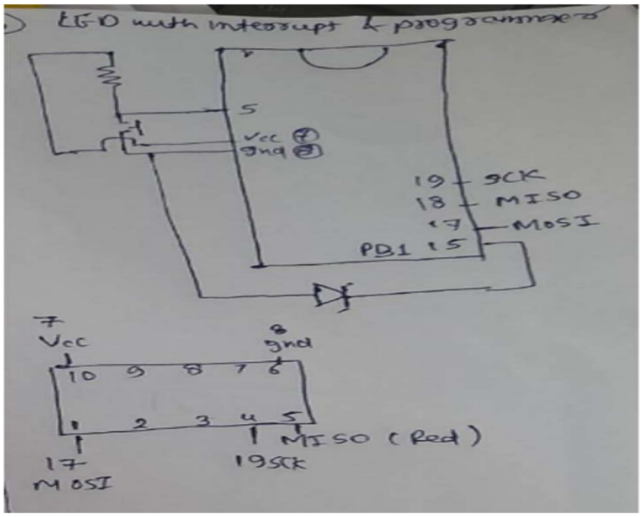


Figure 1: Microcontroller to In-System Programmer Connections



CIRCUIT

1.Interrupts using Assembly level programming

Use int0. Once the switch is pressed the LED should blink 10 times (ON (or OFF) - 1 sec, duty cycle could be 50%).

Code

```
#include "m8def.inc"

.org 0;

rjmp reset;


.org 0x0002;

rjmp int1_ISR;


.org 0x0100;


reset:

    LDI R16,0x70;

    OUT SPL,R16;


    LDI R16,0x00;

    OUT SPH,R16;


    LDI R16,0x01;

    OUT DDRB,R16;
```

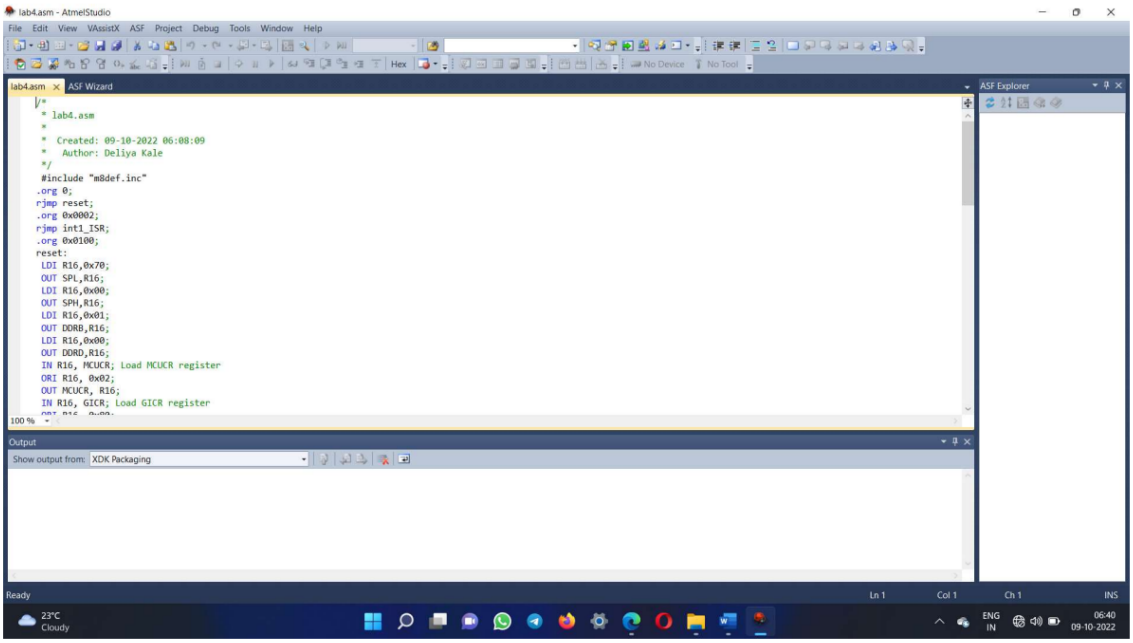
```
        LDI R16,0x00;
        OUT DDRD,R16;

        IN R16, MCUCR;
        ORI R16, 0x02;
        OUT MCUCR, R16;

        IN R16, GICR;
        ORI R16, 0x80;
        OUT GICR, R16;

         LDI R16, 0x00;
         OUT PORTB, R16;
         SEI;

Ind_loop:
        RJMP ind_loop;
int1_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 Execution

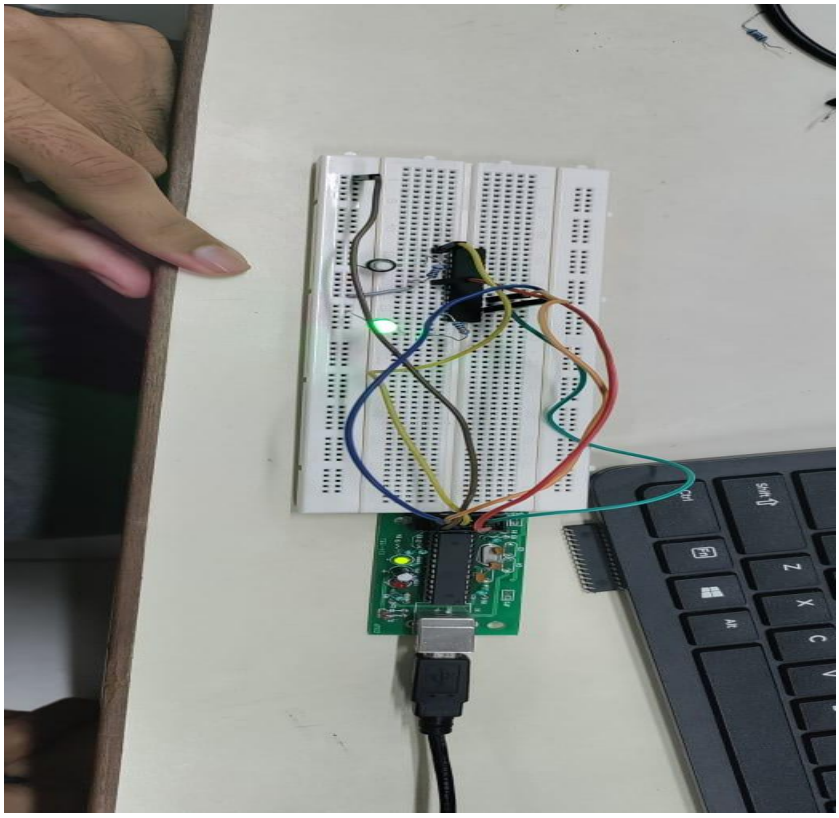


Explanation

The external interrupts are triggered by the INT0, and INT1 pins. Observe that, if enabled, the interrupts will trigger even if the INT0..1 pins are configured as outputs. 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. 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, 8 described in Clock Systems and their Distribution on page 25 in the AVR (Atmega8L) manual uploaded in moodle. 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.

CIRCUIT



Inference

The interrupt programme was executed using Assembly code and the LED blinking for 10 times was achieved at required rate. LED is connected to pin 14 which is the output.

Output video link :

https://drive.google.com/file/d/1otuqdRZf6YSvh9NjLemJqFuiUCdglzjh/view?usp=sharing

2. Interrupt using C code

Perform the same experiment using C program

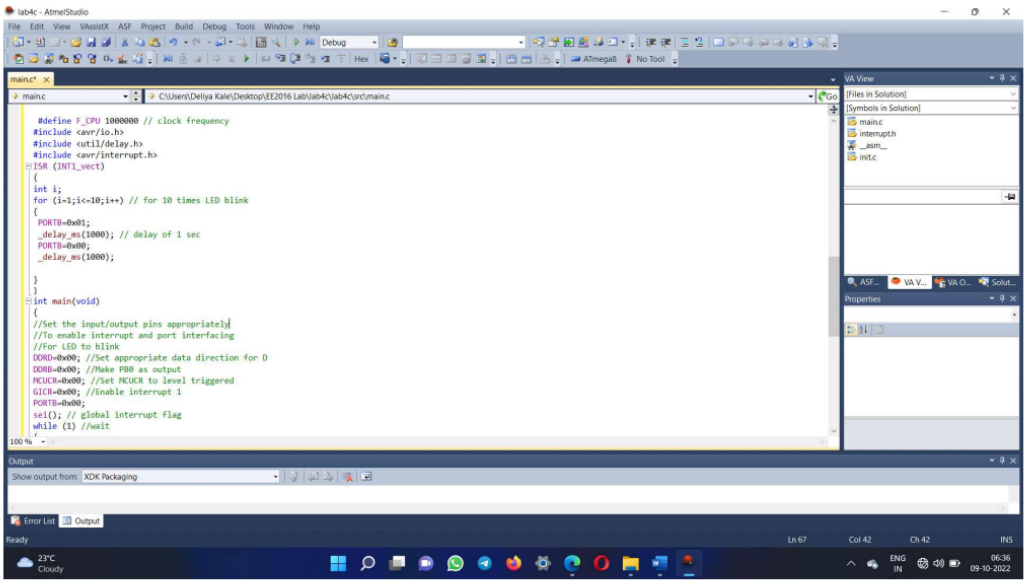
Code

```
#define F_CPU 1000000 // clock frequency

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>


 ISR (INT1_vect)

 {

    int i;
```

```
    for (i=1;i<=10;i++) // for 10 times LED blink
    {
        PORTB=0x01;
        _delay_ms(1000); // delay of 1 sec
       PORTB=0x00;
       _delay_ms(1000);
   }
}
int main(void)
{
     //Set the input/output pins appropriately
    //To enable interrupt and port interfacing
   //For LED to blink
   DDRD=0x00; //Set appropriate data direction for D
   DDRB=0x00; //Make PB0 as output
   MCUCR=0x00; //Set MCUCR to level triggered
   GICR=0x00; //Enable interrupt 1
   PORTB=0x00;
   sei();  // global interrupt flag

   while (1) //wait
   {
```

```
    }
}
```

Code Execution

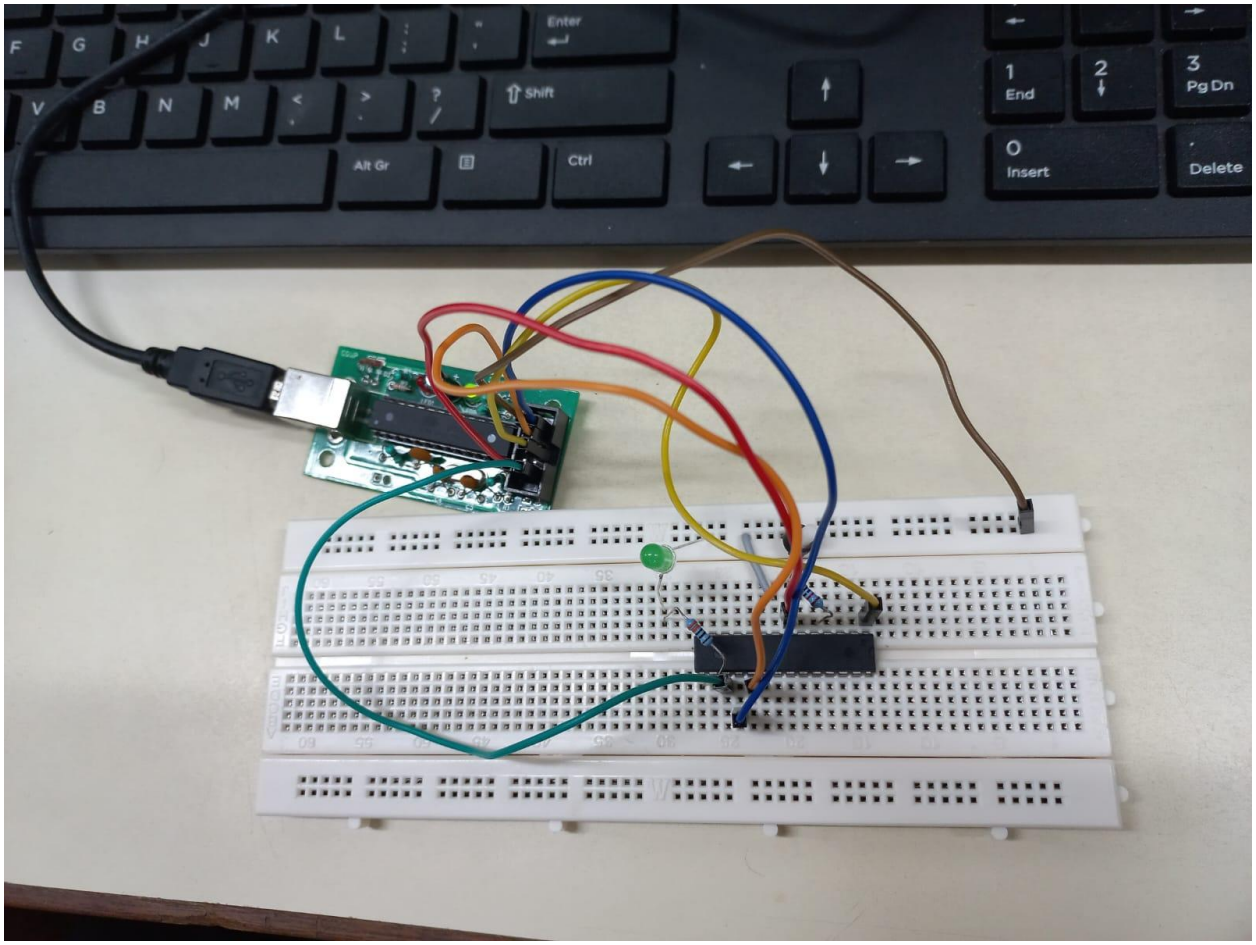


Explanation

The external interrupts are triggered by the INT0, and INT1 pins. Observe that, if enabled, the interrupts will trigger even if the INT0..1 pins are configured as outputs. 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. 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,

described in Clock Systems and their Distribution on page 25 in the AVR (Atmega8L). 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.

CIRCUIT



Output video link :

https://drive.google.com/file/d/1otuqdRZf6YSvh9NjLemJqFuiUCdglzjh/view?usp=sharing

Inference

The interrupt program was executed using C code and the LED blinking 10 times was achieved at the required rate. LED is connected to PIN 14 which is the output