

EXPERIMENT-3 REPORT

KARNIKA SAMYUKTHA C U - EE24B114

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 (Interrupt Service Routine) 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 (via interrupt) to make an LED blink with a duration of 1 second.

APPARATUS REQUIRED

1. Atmel Atmega8 Microcontroller chip, USBASP programmer
2. Bread board with hardware components, data / power cables, LED, mini push buttons etc
3. A PC with Microchip/ Studio simulation software loaded and AVR Burn-o-mat software

IMPLEMENTATION OF PROGRAMS

1. Program 1 – Enabling INT1

- Objective: To write an AVR assembly language program that uses an **external interrupt (INT0)** to blink an LED connected to **PORTB** ten times whenever a push button (connected to **PORTD(5th pin)**) is pressed.
- Setup:
 - i. LED connected to PORTB.0
 - ii. Push button connected to PORTD.3 (INT1 pin).
- Program:
 - i. The program initializes the stack pointer, configures **PORTB.0** as an output for an LED, and sets **PORTD.3** as an input for a push button. It then enables the **external interrupt INT1** on the falling edge of the button signal and enables global interrupts.
 - ii. In the main loop, the program waits indefinitely. When the push button is pressed, the **INT1 interrupt triggers**, and the **ISR (Interrupt Service Routine)** executes. Inside the ISR, the LED connected to PORTB.0 blinks **10 times**, using software

delay loops to make the ON and OFF periods visible. After completing 10 blinks, the ISR restores registers and returns control to the main loop, where it waits for the next button press.

- Observation:
 - Global interrupts and external interrupt INT0 are enabled.
 - When the **push button** is pressed, INT0 interrupt triggers:
 - The ISR executes a loop to blink the LED on PORTB.0 exactly **10 times**.
 - Simple delay loops (a1, a2, b1, B2) are used for visible blinking.
 - After completing 10 blinks, the ISR returns, and the program waits for the next button press.

Video Link:

https://drive.google.com/file/d/1vvCN5jL6OLCRcQNeFJSxRgGBYUfKV_4V/view?usp=drive_link

```
;Author: ee24b114_ee24b115

.org 0; this sets the pc at address 0x0000(reset vector)
rjmp reset
.org 0x0002; interrupt vector address of external interrupt (inte) org 0x0100; program starts at program memory address 0x0100(conventional)
rjmp int1_ISR
reset:

LDI R16,0x70
OUT SPL, R16;stack pointer low
LDI R16,0x00
OUT SPH, R16; stack pointer high
;for LED connections
LDI R16, 0x01
OUT DDRB, R16

;for push button
LDI R16, 0x00
OUT DORD, R16

IN R16, MCUCR

ORI R16, (1<<ISC11); ISC11 bit configures the interrupt sense of control for int0 setting ISC11=1(and ISC00=0), means int1 triggers on the falling edge
OUT MCUCR, R16

IN R16,GICR
ORI R16, (1<<INT1); enables external inti(this activates int0 ISR when the button is pressed)

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; to get 10 blinks
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

```

2. Program 2 – Enabling INT0

- Objective: To write an AVR assembly language program that uses an **external interrupt (INT0)** to blink an LED connected to **PORTB** ten times whenever a push button (connected to **PORTD(4th pin)**) is pressed.
- Setup:
 - i. _LED connected to PORTB.0
 - ii. Push button connected to PORTD.2 (INT0 pin).
- Program:
 - i. The program initializes the stack pointer, configures **PORTB.0** as an output for an LED, and sets **PORTD.2** as an input for a push button. It then enables the **external interrupt INT0** on the falling edge of the button signal and enables global interrupts.
 - ii. In the main loop, the program waits indefinitely. When the push button is pressed, the **INT0 interrupt triggers**, and the **ISR (Interrupt Service Routine)** executes. Inside the ISR, the LED connected to PORTB.0 blinks **10 times**, using software delay loops to make the ON and OFF periods visible. After completing 10 blinks, the ISR restores registers and returns control to the main loop, where it waits for the next button press.
- Observation:
 - Global interrupts and external interrupt INT0 are enabled.
 - When the **push button** is pressed, INT0 interrupt triggers:
 - The ISR executes a loop to blink the LED on PORTB.0 exactly **10 times**.
 - Simple delay loops (a1, a2, b1, B2) are used for visible blinking.
 - After completing 10 blinks, the ISR returns, and the program waits for the next button press.

Videolink: <https://drive.google.com/file/d/1vtbZekkMyeerMNDIW1Sz5A65r6EmjJw/view?usp=sharing>

```
};
;Author: ee24b114_ee24b115

.org 0; this sets the pc at address 0x0000(reset vector)
rjmp reset
.org 0x0001; interrupt vector address of external interrupt (inte) org 0x0100; program starts at program memory address 0x0100(conventional)
rjmp int0_ISR
reset:

LDI R16,0x70
OUT SPL, R16;stack pointer low
LDI R16,0x00
OUT SPH, R16; stack pointer high
;for LED connections
LDI R16, 0x01
OUT DDRB, R16

;for push button
LDI R16, 0x00
OUT DORD, R16

IN R16, MCUCR

ORI R16, (1<<ISC01); ISC11 bit configures the interrupt sense of control for int0 setting ISC01=1(and ISC00=0), means int1 triggers on the falling edge
OUT MCUCR, R16

IN R16,GICR
ORI R16, (1<<INT0); enables external inti(this activates int0 ISR when the button is pressed)

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; to get 10 blinks
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
```