Experiment 4: Interrupts in Atmel AVR Atmega through Assembly Programming

Abhinav I S EE23B002

September 17, 2024

1 Objectives

1. Generate an external hardware interrupt using a push button switch
2. Write an Interrupt Service Routine to switch ON an LED for a few seconds, and then switch OFF.
3. Achieve all of this in AVR Assembly

2 Procedure

2.1 Code

```
;The interrupt pin chosen to be used is INT0 = PD2
.include "m8def.inc"

.def ledR = r17
.def ledtrig = r16
.def outloopR = r18
.def inloopRL = r24
.def inloopRH = r25
.def mloopR = r20; This is used to control the led blinking

.equ inloopC = 1760;
.equ outloopC = 71; These two together will make the led with a frequency of 1Hz
.equ mloopC = 20; this makes sure the led toggles 20 times, for a total of 10 secs

.cseg
.org 0
rjmp reset; Jumps the interrupt vector table to the main program

.org 0x02
rjmp isr; Calls the ISR when the interrupt is triggered

.org 0x0100
reset:
  clr ledR
  ldi ledtrig, 0x01
  out DDRB, ledtrig; Sets port B0 as output for the led to be triggered
  ldi r31, 0x00
  out PORTB, r31
  ldi r19, HIGH(RAMEND)
```

```
  out SPH, r19
  ldi r19, LOW(RAMEND)
  out SPL, r19; Sets the stack pointer to hold the ISR
  sbi PORTD, 2
  ldi r19, 0x00; Sets portD as input only, which can then trigger the interrupt
  out DDRD, r19

  IN R30, MCUCR;Load MCUCR register
  ORI R30, 0x00;
  OUT MCUCR, R30

  ldi r19, 1<<INT0; can enable int0
  out GICR, r19; enables the interrupt pin
  sei; enables the interrupts

forever:
  rjmp forever ; puts the microcontroller in a loop waiting for the interrupt

isr: ; isr, which accomplishes the led blink as required
  clr ledR
  mloop:
    ldi mloopR, 20

  l1:
    ldi ledtrig, 0x01
    eor ledR, ledtrig
    out PORTB, ledR
    ldi outloopR, outloopC
  l2:
    ldi inloopRH, HIGH(inloopC)
    ldi inloopRL, LOW(inloopC)

  l3:
    sbiw inloopRL, 1
    brne l3

    dec outloopR
    brne l2

    dec mloopR
    brne l1

    ldi r31, 0x00
    out PORTB, r31 ; turns the led off after the ISR execution
    reti : return of control to the previously executing process
```

2.2 General Understanding About Interrupts

While an interrupt is activated, the microcontroller goes through the following steps:

1. Finishes executing the current instruction, and then stores the address of the next instruction on the stack.
2. It jumps to a dixed location in memory (interrupt vector table). The interrupt vector table directs the AVR microcontroller to the address of the interrupt service routine (ISR)

3. Upon executing the RETI instruction, the microcontroller goes back to the address from where the interrupt was originally triggered.

- from Mazidi, page 363

2.3 Explanation of code

1. The .def directives alias the registers for storing different values, for better code readability
2. Here, ledtrig holds the I/O port of the led to be triggered.
3. ledR, is the register where we will hold the value to be XORed with the I/O port.
4. outloop, inloop are the registers we use to store the loop iteration variabes.
5. .org 0 is the beggining of the program,
6. Here we jump to reset, when the program starts, skipping the interrupt vector table
7. And at 0x02, which is the location of the External Interrupt request 0 in the Interrupt Vector Table, here, we jump to isr, which is the ISR that we wrote.
8. moving onto the reset subroutine:
   (a) we clear the ledR
   (b) we make sure the LED remains OFF.
   (c) We setup pinB0 to be output, hence (0x01) in the reg DDRB
   (d) We set up the stackspace.
   (e) We set the port D as input only, so that we can trigger the interrupt.
   (f) We set the 2nd bit becaues int0 corresponds to pin PD2.
   (g) We can enable int0, in the GICR
   (h) We enable interrupts using the sei instruction
9. The forever routine, just keeps the microcontroller in a loop, that does nothing, waiting for an interrupt to be triggered
10. the ISR(Interrupt Service Routine)
    (a) Consists of an mloop, that decrements down from 20 (for 10 led blinks)
    (b) The l2, l3 (outloop, and inloop) creates the delay, between two LED blinks.
    (c) After each decrement of mloop, we toggle the led, and at the end after making sure the LED is off, we execute reti, to go back to the forever loop.
    (d) the delay was calculated, similar to the last AVR experiment, by taking into account the clock frequency of the microcontroller (1Mhz) and the number of clock cycles taken for each instruction.

2.4 Snapshot of the circuit.

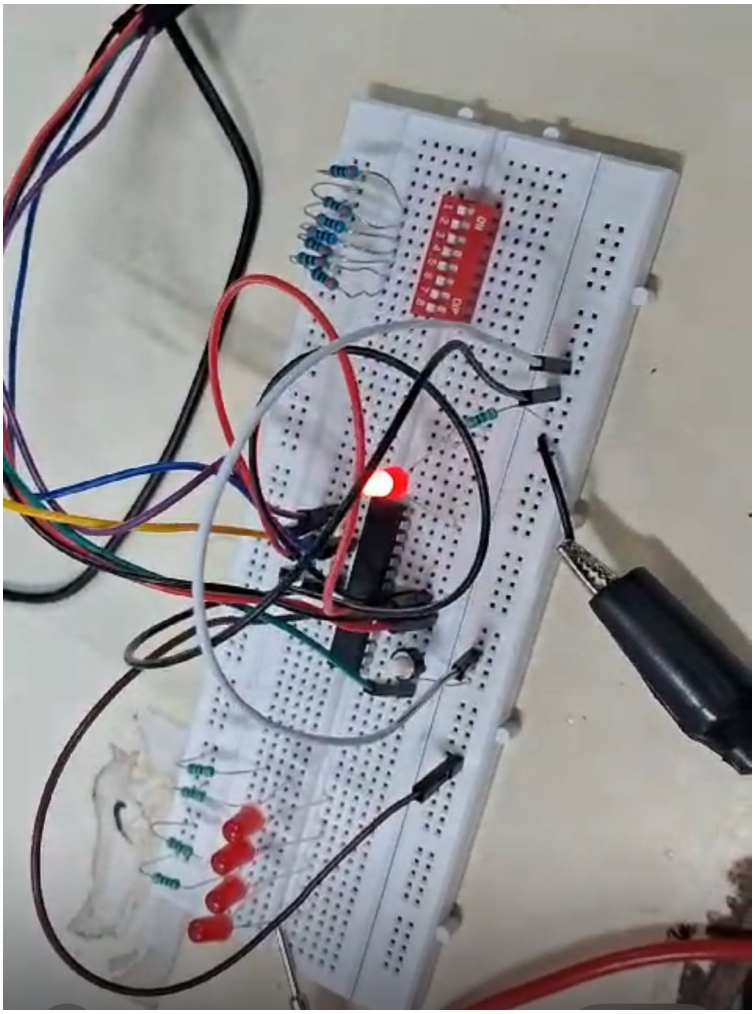

Figure 1: AVR Interrupt

2.5 My contribution

I helped out in figuring out the time delay part of the ISR, and the reset subroutine, apart from the wiring.