**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Dr. Arbab Latif  **Dated:** 02/04/2022

**Semester:** 4th **Section:** BEE 12C

**EE-222: Microprocessor Systems**

Lab 7: AVR Interrupts

Group Members

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Reg. No** | **Report**  **Marks / 10** | **Viva**  **Marks / 5** | **Total**  **Marks / 15** |
| Danial Ahmad | 331388 |  |  |  |
| Muhammad Umer | 345834 |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Experiment

# Functions, Branches and Delays

## Objectives

1. Control GPIOs of ATmega16A
2. Implement branches in assembly language
3. Implement calculated delays
4. Break the code down to modular functions

## Equipment

Software

* *Atmel Studio*

Hardware

* ATmega16A microcontroller unit
* Universal Programmer
* Seven Segment Display
* Resistance 47Ω
* LEDs (may use from trainer kit)
* Switch or Button (may use from trainer kit)

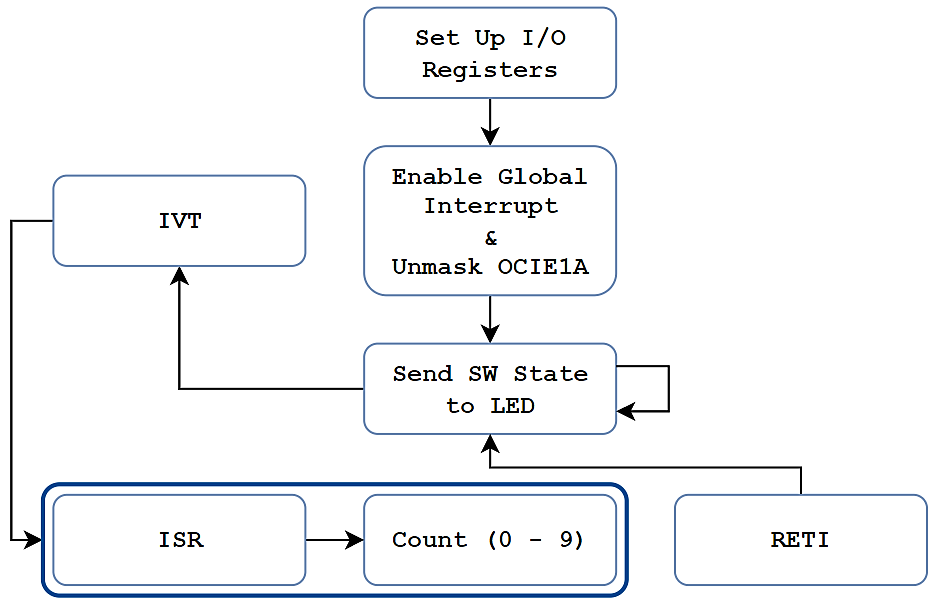
## Introduction

 Atmel Studio IDE is a free development environment for programming Atmel MCUs, sourced by Microchip Technology Inc. It provides us with the means to simulate assembly language codes on specific Microcontrollers and provides an easy and intuitive way of producing .HEX files, which are what makes burning the code on the hardware possible.

In this specific lab, we familiarize ourselves with AVR programming through interrupts and observing its advantages over polling method. Interrupts allow us to perform two separate tasks in parallel rather than being stuck in a loop, unable to utilize MCU’s resources anywhere, as is the case in polling method.

# Lab Tasks

## Flowchart



## Task A

1. Connect a seven segment display “SSEG” with your ATmega16A.
2. Connect an led “LED” and a switch “SW” with your ATmega16A.
3. The SSEG should display continuous counting from 0-9 and then again from 0, each digit with a 1 second delay.
4. At the same time, the state of SW should be relayed to LED (–that is, if SW is high, LED should be on and vice versa).
5. Note that both of the tasks are completely independent and should be performed in parallel without lag.

**Code**

**#include <avr/io.h>**

**#include <avr/interrupt.h>**

**char SevenSegment (char); // Function Prototype**

**#define LED 0**

**char counter = 0;**

**int main(void)**

**{**

**DDRA = 0x00; // PORTA (Utilizing only PA0) as INPUT**

**DDRB = 0x01; // PB0 as OUTPUT**

**DDRC = 0xFF; // PORTC as OUTPUT**

**TCNT1H = 0x00; // Initialize Counter**

**TCNT1L = 0X00;**

**TCCR1A = 0x00; // CTC Mode, Prescaler: 64**

**TCCR1B = 0x0B;**

**OCR1A = 0x3D09; // Output Compare Register**

**sei(); // Enable Global Interrupt**

**TIMSK = 0x10; // OCIE1A Unmasked**

**while (1) {**

**if( (PINA & 0X01) == 1) // If PA0 is HIGH**

**{**

**PORTB = 1 << LED; // Roll over 1 to LED Position**

**}**

**else // If PA0 is LOW**

**{**

**PORTB = 0 << LED; // Roll over 0 to LED Position**

**}**

**}**

**}**

**ISR(TIMER1\_COMPA\_vect) // TIMER1 OCR1A ISR**

**{**

**PORTC = SevenSegment(counter);**

**counter += 1;**

**if (counter == 10)**

**{**

**counter = 0;**

**}**

**TIFR = 1 << OCF1A; // Output COMPARE Flag Cleared.**

**}**

**// ------- BCD - 7 Segment Decoder -------**

**char SevenSegment(char number)**

**{**

**if (number == 0) return 0x01;**

**else if (number == 1) return 0x4F;**

**else if (number == 2) return 0x12;**

**else if (number == 3) return 0x06;**

**else if (number == 4) return 0x4C;**

**else if (number == 5) return 0x24;**

**else if (number == 6) return 0x20;**

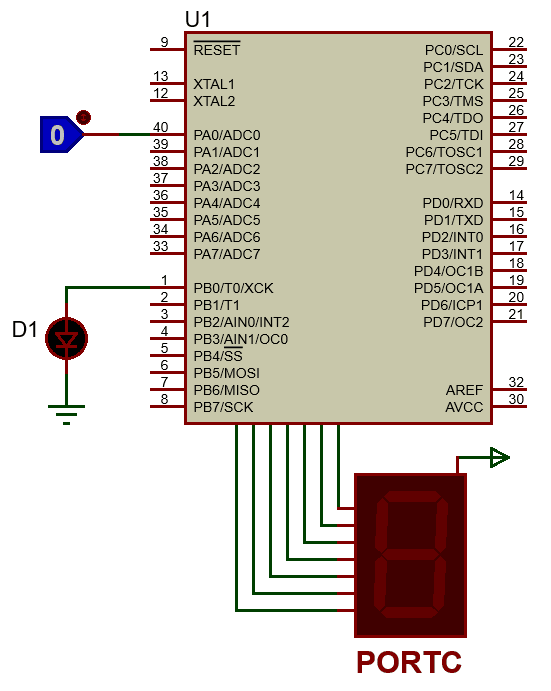
**else if (number == 7) return 0x0F;**

**else if (number == 8) return 0x00;**

**else if (number == 9) return 0x0C;**

**else return 0xFF;**

**}**



**Proteus Simulation**

## Conclusion

After the conduction of this lab, we have learnt how to utilize interrupts in AVR and implemented a program that performs two distinct tasks; sends data from one PORT to another and counts up from 0 – 9, in parallel and without any lag. Such techniques allow us to allocate more of MCU’s resources in a given amount of time and accomplish more tasks rather than being unoptimal as is the case in polling method.