**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Dr. Arbab Latif  **Dated:** 14/03/2022

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

**EE-222: Microprocessor Systems**

Lab 5: Timer Programming

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)

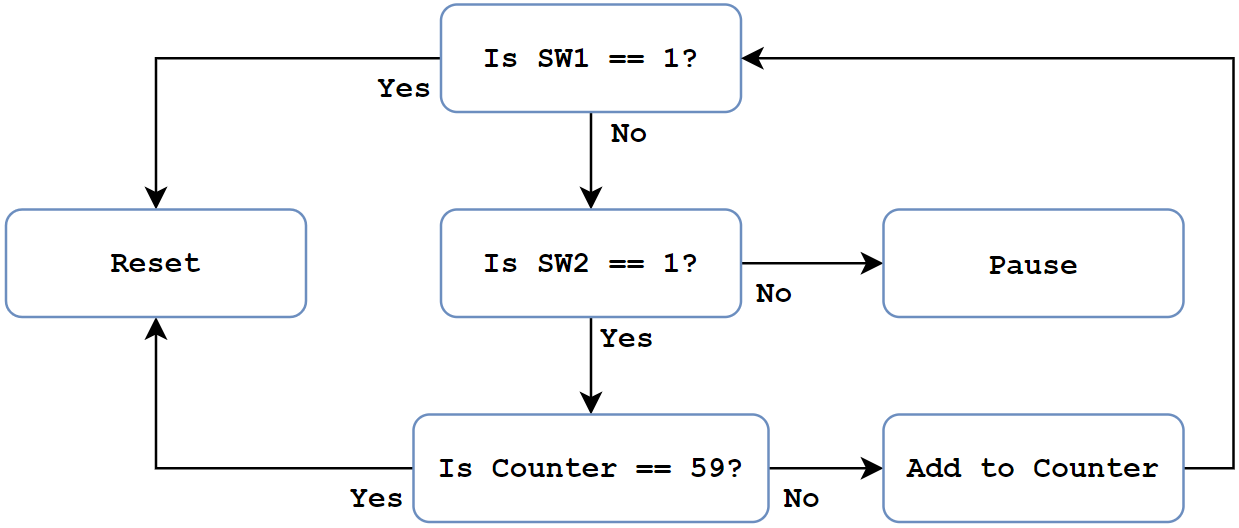
## Introduction

 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 in C language as well as enhancing the applicability of delays through the use of timers. We interface two BCD – 7 Segment displays to create a stopwatch with each increment having a delay of 1s, with reliable accuracy. Common observations include a much concise and a human – readable code relative to programming purely in assembly language.

# Lab Tasks

## Flowchart



## Task A

Implement the code in example 2.5.5, just to get familiar with C programming.

**Code**

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

**int main()**

**{**

**DDRB = 0x00; // Set port B as input**

**DDRC = 0xFF; // Set port C as output**

**TCNT1 = 0; // Initialize timer 1**

**OCR1A = 62500; // Set output compare register**

**TCCR1A = 0x00; // WGM13:WGM10 = 0100, CS12:CS10 = 010**

**TCCR1B = 0x0A;**

**unsigned char data;**

**while(1)**

**{**

**if( (TIFR & 0x10) != 0 ) // Check OCF1A flag bit**

**{**

**TIFR |= 0x10; // Clear flag**

**data = PINB; // Read port B**

**if( (data & 0x01) == 0)**

**{**

**PORTC = 0b01110001;**

**}**

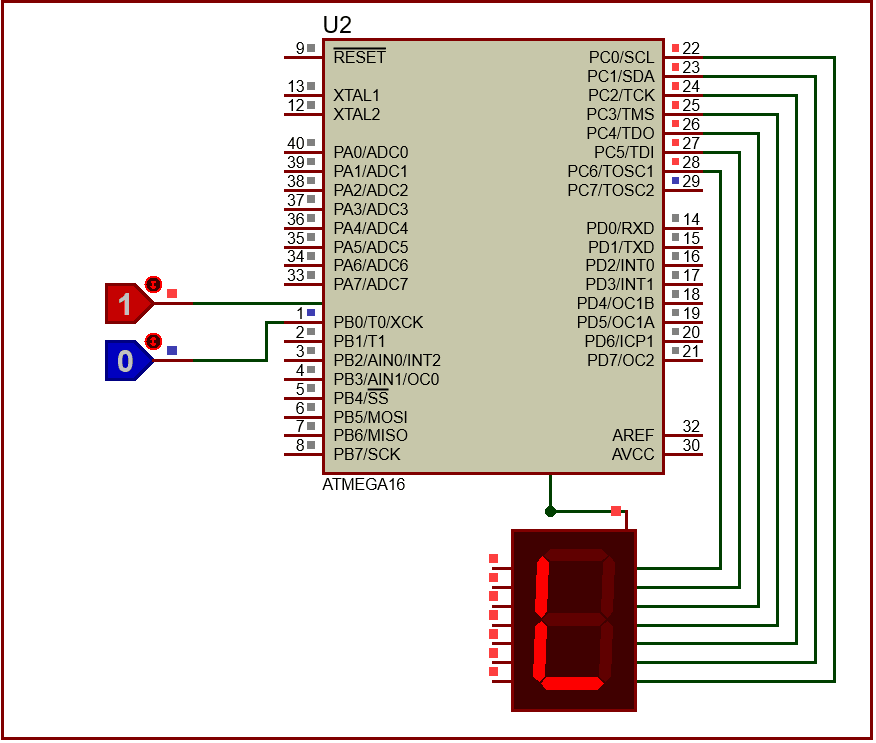
**else**

**PORTC = 0b01001000;**

**}**

**}**

**}**



## Task B

Calculate and determine the value of compare match and prescalar for timer 1 such that it produces a precise 1 second delay for 1MHz clock. Include your calculations in the report*.*

**Time for one tick = 64/1M = 64µs**

**Ticks for 1000ms = 1000m/64µ = 15625 = 0x3D09**

## Task C

In this task you are required to create a “Digital Stop Watch” that records the time in seconds precisely (use CTC mode).

1. Connect two 7-segment-Displays with your ATmega16A.
2. Connect two switches (say Sw1 and Sw2) with your ATmega16A.
3. If Sw1 is high, the Stop Watch must get reset to zero, no matter what is the state of Sw2.
4. If Sw2 is high and Sw1 is low, the Stop Watch must display the seconds passed.
5. If Sw2 is low and Sw1 is also low, the Stop Watch must pause its time and if Sw2 is raised again, it should resume from where it was paused

**/\***

**\* Lab5.cpp**

**\***

**\* Created: 14/03/2022 8:13:34 pm**

**\* Author : Danial and Umer**

**\*/**

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

**char SevenSegment(char number);**

**int main()**

**{**

**DDRA = 0x00; // Set port A as input**

**DDRB = 0xFF; // Set port B as output**

**DDRC = 0xFF; // Set port C as output**

**TCNT1 = 0; // Initialize timer 1**

**OCR1A = 0x3D09; // Set output compare register**

**TCCR1A = 0x00; // WGM13:WGM10 = 0100 (CTC), CS12:CS10 = 011**

**TCCR1B = 0x0B;**

**// unsigned char data;**

**unsigned char counter = 0;**

**while (1) {**

**if (PINA > 1) // Reset**

**{**

**counter = 0;**

**PORTB = SevenSegment(counter / 10);**

**PORTC = SevenSegment(counter % 10);**

**}**

**else if (PINA == 1) // play**

**{**

**if ((TIFR & 0x10) != 0) // Check OCF1A flag bit**

**{**

**if (counter == 59) // If Counter == 59, then Reset**

**{**

**counter = 0;**

**PORTB = SevenSegment(counter / 10);**

**PORTC = SevenSegment(counter % 10);**

**TIFR |= 0x10; // Clear flag**

**}**

**else // Normal Up counting**

**{**

**counter += 1;**

**PORTB = SevenSegment(counter / 10);**

**PORTC = SevenSegment(counter % 10);**

**TIFR |= 0x10; // Clear flag**

**}**

**}**

**}**

**else if (PINA == 0) // Pause**

**{**

**// Do no Operation**

**}**

**}**

**}**

**// ------- 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;**

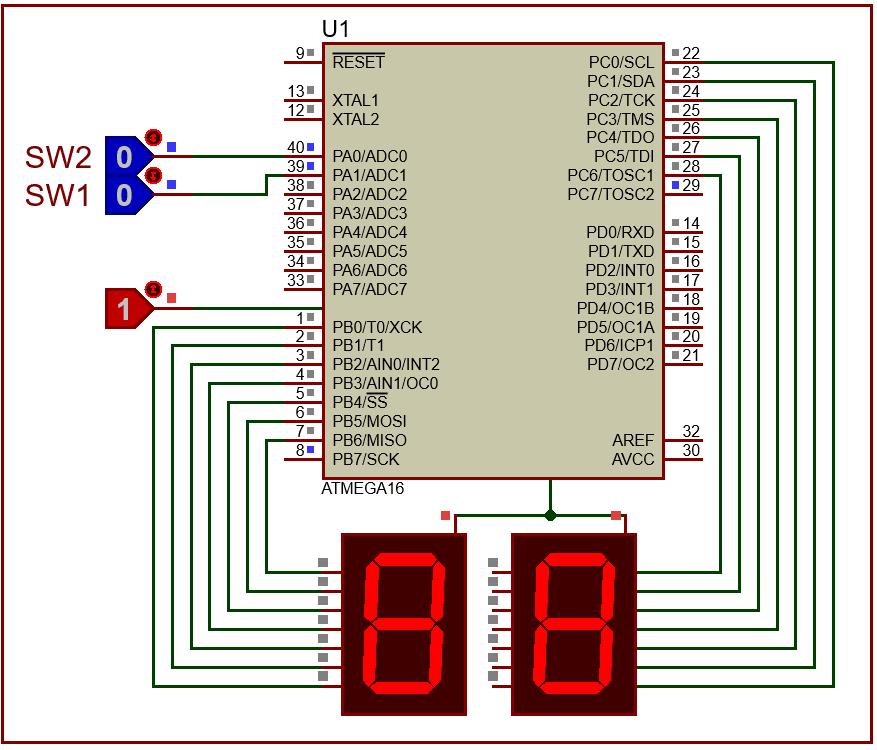
**else if (number == 7) return 0x0F;**

**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 the timer registers in AVR and implemented a stop watch through means of a simple program and two BCD – 7 Segment decoders. We observe that implementing delays in such a way results in a concise and much readable code, but perhaps the biggest advantage of timers is the accuracy in delays relative to coding through nested loops.