

**Namal University**

**Mianwali**

**Department of Electrical Engineering**

EE-252L: Introduction to Embedded Systems

**Lab Manual: 08**

**AVR Programming in C Language I/O Operations and Timer Introduction**

|  |  |
| --- | --- |
| **Students Name** | **Fahim Ur Rehman Shah** |
| **Roll Number** | **NIM-BSEE-2021-24** |
| **Submission Date** | **6-8-2023** |
| **Marks Obtained** |  |

**Instructors: Dr. Hamza Zad Gul**

# Objectives

# In this lab, the student will learn about logic programming and I/O port interfacing to configure the programmer and Microchip studio to program atmega328p.

# Course Learning Outcomes

CLO1: Practice the correct use of programming constructs of assembly language

CLO2: Construct systems by interfacing AVR peripherals

CLO3: Perform the assigned task individually/as a team effectively

CLO4: Report the outcomes of task performed effectively in oral and written form

# Software

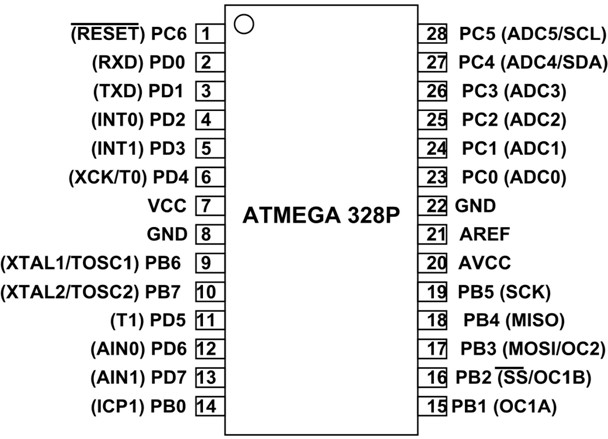
* Microchip studio

# Hardware

* Atmega 328p
* Atmega328p USBasp programmer Board
* Breadboard
* Connecting wires
* LEDs
* Resistors
* Capacitors
* Crystal oscillator
* Push buttons
* Oscilloscope

# Instructions

* You must submit the lab report complete within given deadline.
* Plagiarism or any hint thereof will be dealt with strictly. Any incident where plagiarism is caught, both (or all) students involved will be given zero marks, regardless of who copied whom.
* Multiple such incidents will result in disciplinary action being taken.



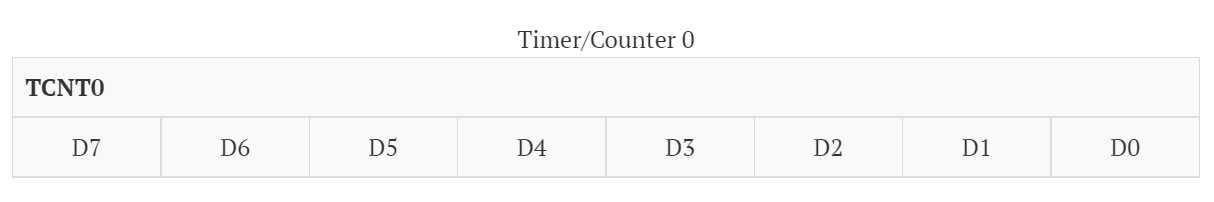
**Introduction:**

**Timers/Counters:**

Timers/counters modules are used to generate precise time delay. They are separate hardware in a microcontroller which operates independently of microcontrollers. The Atmega328p has three timers timer0, timer1, timer2. These can be configured in different modes to generate precise delays and PWM signal.

**Timer0 (Normal Mode):**

The goal is to generate 500ms delay with timer zero. The timer0 is an 8-bit timer which means it can count up to 255. The operation of timer0 is pretty straight forward it increment TCNT0 register on every clock cycle. TCNT0 is an 8-bit register and holds the timer count. When it reaches up to the value of 255 then roll back timer count and sets the timer overflow flag(TOV). To run a timer from a specific value, we can use output compare registers (OCR0).



The configuration of timer is controlled by TCCR0b register which have different flag. These flags can be set to achieve the desired behavior. For normal mode we need to set following two flags.

1. The Frequency of the Clock Source with CS02, CS01, CS00 bits.

2. The mode of the timer.

Table

Description automatically generated

Table

Description automatically generated

The Timer/counter Interrupt Flag Register (TIFR) holds the two basic flags we need the TOV and OVF.

Table

Description automatically generated with medium confidence

## Lab Task I

1. Go to File 🡪 New 🡪 Project as shown below

Graphical user interface, text, application

Description automatically generated

1. Select C/C++ and GCC C Executable Project

Graphical user interface, application

Description automatically generated

1. Select the Device you are using.

Graphical user interface, application

Description automatically generated

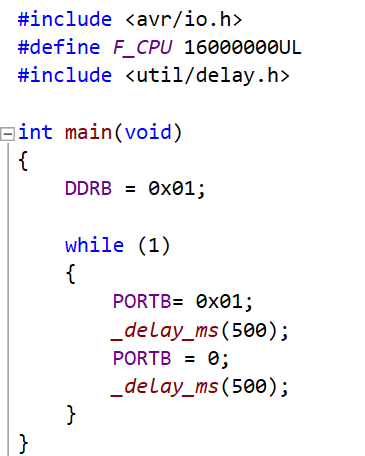
1. The project will look like this

Graphical user interface, text, application, email

Description automatically generated

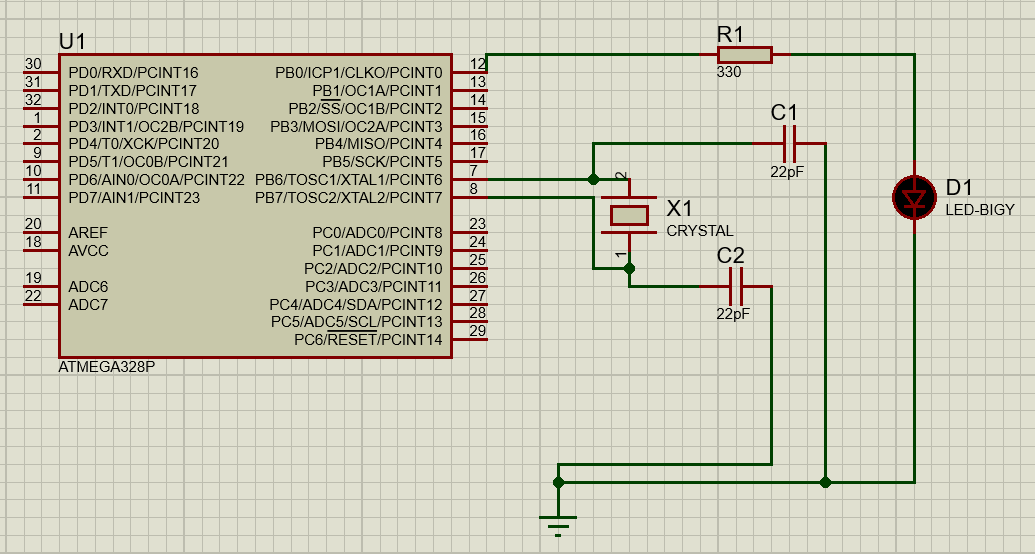
1. You can start writing your code.

**Lab Task 1: Write the following code in main.c file.**



**Lab Task 2: Test your code on Proteus:**

Construct the following circuit in proteus and hardware. Run your code and show it to instructor.



## Make sure following setting is provided when you double click the microcontroller in proteus.

## 

## Lab Task 3:

## Modify the above code so that when a push button at PC0 send a LOW signal the LED at PB1 turns ON otherwise the LED is OFF. Write the code in microschip studio and implement the circuit on proteus and show the working to the lab instructor. Also implement this on hardware. Attach your code, proteus circuit and breadboard implemented circuit below

## Code:

/\*

\* Lab\_08\_Task\_03.cpp

\*

\* Created: 6/6/2023 1:46:23 PM

\* Author : fahim

\*/

#include <avr/io.h> // Include the AVR IO library for accessing registers and I/O operations

#define *F\_CPU* 16000000UL // Define the CPU frequency

int main(void)

{

DDRB = 0b00000010; // Set PORTB pin 1 as output

DDRC = 0x00; // Set PORTC as input

while(1) // Infinite loop to continuously monitor the input pin

{

if(PINC & 0b00000001) // If the least significant bit of PINC is high

{

PORTB = 0x00; // Set PORTB pin 1 to low (turn off the LED)

}

else

{

PORTB = 0b00000010; // Set PORTB pin 1 to high (turn on the LED)

}

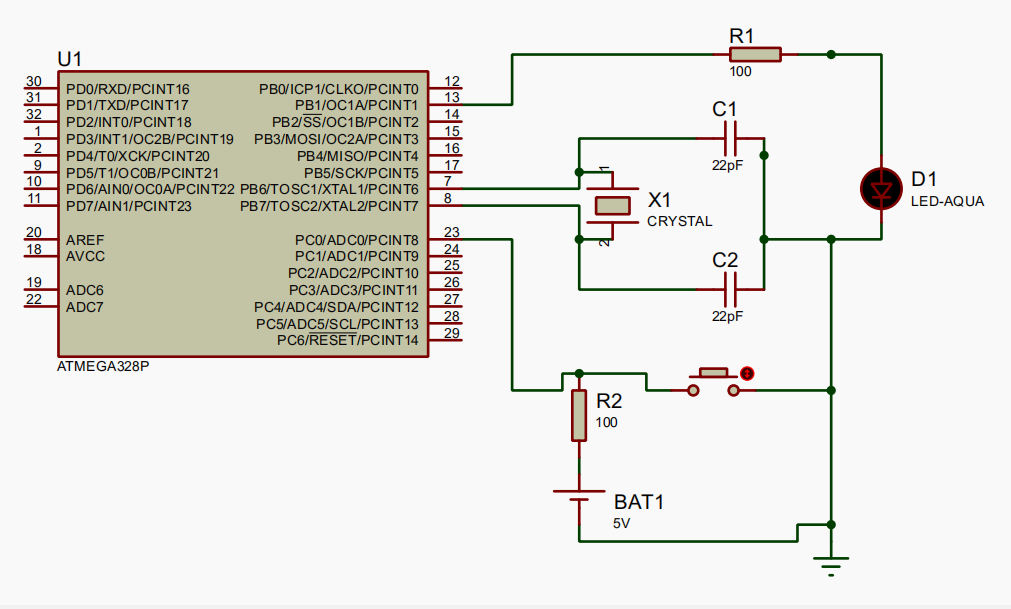
}

return 0;

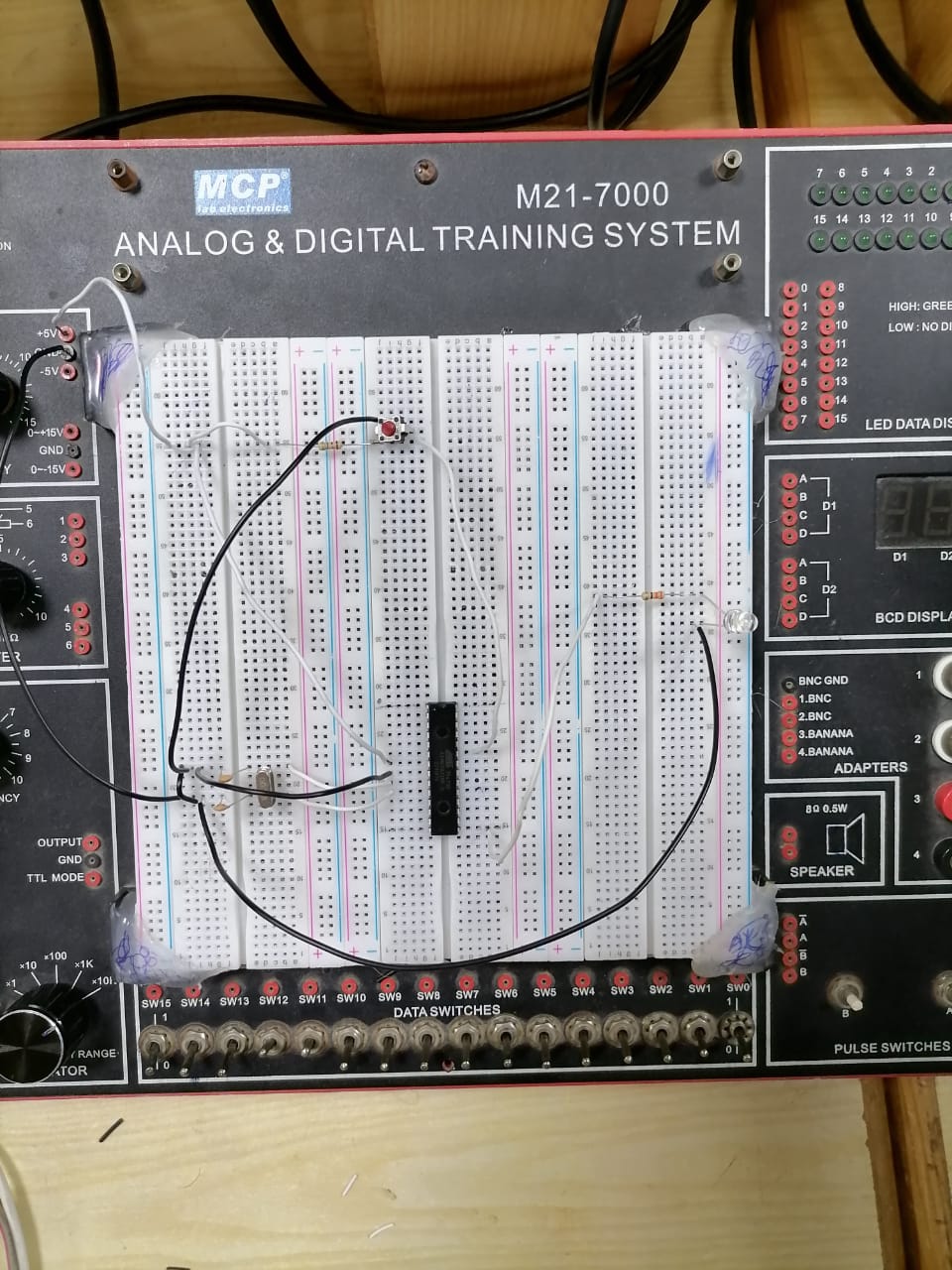
}

***Circuit :***

1. ***Simulation ( Proteus ) :***



1. ***Hardware ( Breadboard ) :***

******