



ECA14

EMBEDDED SYSTEM



Engineer to Excel

LAB RECORD

DEPARTMENT OF ECE

SAVEETHA SCHOOL OF ENGINEERING



SAVEETHA UNIVERSITY

CHENNAI 602 105

INDEX

S.NO.	DATE	EXPERIMENT	SIGNATURE
		UNIT - 1	
1		STUDY OF ARDUINO	
2		BLINKING OF LED USING 8051 MICROCONTROLLER USING KEIL	
3		STEPPER MOTOR INTERFACING WITH 8051 MICROCONTROLLER USING KEIL	
4		DC MOTOR INTERFACING WITH 8051 MICROCONTROLLERS USING KEIL	
5.		A.INTERFACING LED	
		B.INTERFACING PWM	
		UNIT - 2	
6		A.KEYBOARD INTERFACE	
		B.LCD MATRIX KEYBOARD INTERFACE	
7		A.ANALOG TO DIGITAL SIGNAL CONVERSION	
		B.DIGITAL TO ANALOG SIGNAL CONVERSION	
		UNIT - 3	
8		A.SERIAL PORT INTERFACE	
		B.INTERFACING REAL – TIME CLOCK	
9		A.INTERFACING EEPROM	
		B.INTERFACING INTERRUPT	
10		MAILBOX	
		UNIT - 4	
11		BLINKING OF AN LED USING ARDUINO	

12		FADING OF AN LED USING ARDUINO	
13		TURNING AN LED ON AND OFF USING A PUSHBUTTON	
		UNIT - 5	
14		INTERFACING A WATER-LEVEL SENSOR WITH AN ARDUINO	
15		INTERFACING AN ULTRASONIC SENSOR WITH AN ARDUINO	
16		TO USE A BUZZER (OR PIEZO SPEAKER) WITH ARDUINO	
17		MQ-6 GAS SENSOR INTERFACING WITH ARDUINO	
18		IMPLEMENTING ZIGBEE PROTOCOL WITH ARM	
19		A.INTERFACING LCD MODULE WITH ARDUINO	
		B.INTERFACING RFID MODULE WITH ARDUINO	
		EXPERIMENTS BEYOND SYLLABUS	
20		SIMULATION USING PROTEUS SOFTWARE – AN INTRODUCTION	
21		INTERFACING OF CALCULATOR USING 8051 MICROCONTROLLER IN PROTEUS	

Exp. No : 1

STUDY OF ARDUINO

Date:

Aim:

To study the features, working, and architecture of an Arduino.

Component required:

- Arduino UNO

Theory:

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many

others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than \$50
- Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Arduino UNO

The Arduino UNO is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 Digital pins, 6 Analog pins, and programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable. It can be powered by a USB cable or by an external 9 volt battery, though it accepts voltages between 7 and 20 volts.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform. The ATmega328 on the Arduino Uno comes pre-programmed with a bootloader that allows uploading new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol. The Uno also differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Technical specifications:

- Microcontroller: Microchip ATmega328P
- Operating Voltage: 5 Volt
- Input Voltage: 7 to 20 Volts
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 20 mA
- DC Current for 3.3V Pin: 50 mA

- Flash Memory: 32 KB of which 0.5 KB used by bootloader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Speed: 16 MHz
- Length: 68.6 mm
- Width: 53.4 mm
- Weight: 25 g

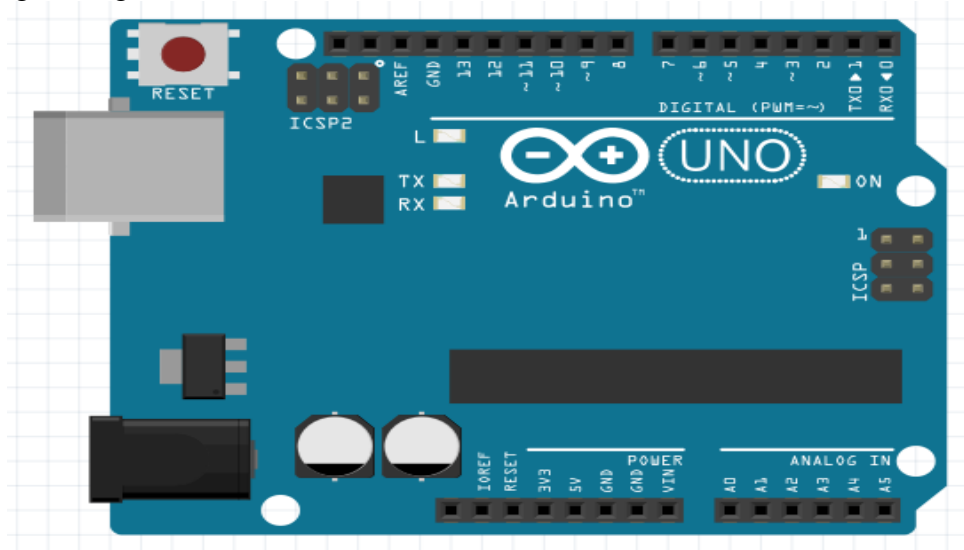


Figure - 2.1: Arduino UNO

Pin Details:

General pin functions

- LED: There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- VIN: The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board.
- 3.3V: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND: Ground pins.
- IOREF: This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.
- Reset: Typically used to add a reset button to shields which block the one on the board.[7]

Special Pin Functions

Each of the 14 digital pins and 6 Analog pins on the Uno can be used as an input or output, using pin Mode (), digital Write (), and digital Read () functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not

be exceeded on any I/O pin to avoid permanent damage to the microcontroller. The Uno has 6 analog inputs, labelled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and the `analogReference()` function.

In addition, some pins have specialized functions:

- Serial: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- PWM(Pulse Width Modulation) 3, 5, 6, 9, 10, and 11 Can provide 8-bit PWM output with the `analogWrite()` function.
- SPI(Serial Peripheral Interface): 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- TWI(Two Wire Interface): A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.
- AREF(Analog REference): Reference voltage for the analog inputs.

Communication

The Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A `SoftwareSerial` library allows serial communication on any of the Uno's digital pins.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino/Genuino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the boot loader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened.

Result: The features, working, and architecture of an Arduino have been studied.

Exp. No.: 2 BLINKING OF LED USING 8051 MICROCONTROLLER AND KEIL C- AT89C51

Date:

Aim:

To write a program to perform blinking of LED using 8051 microcontroller and Keil C-AT89C51.

Components required:

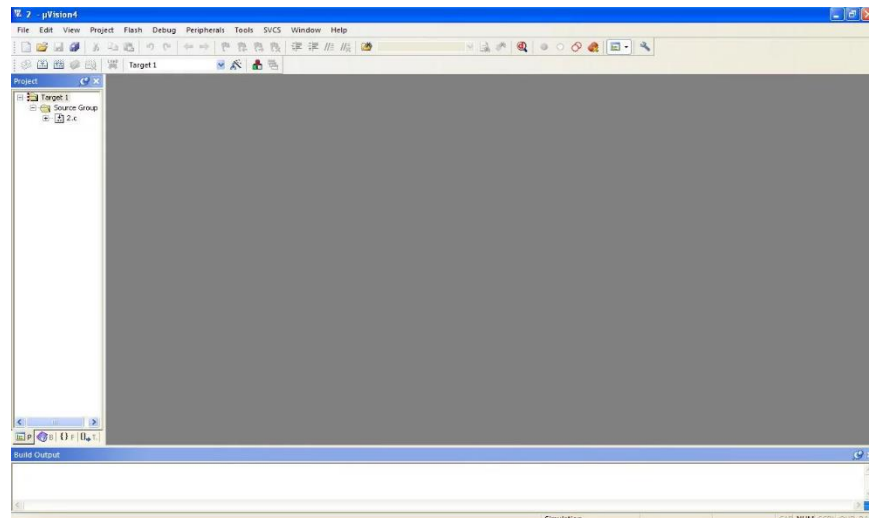
- 8051 Microcontroller
- Keil uVision4

Theory:

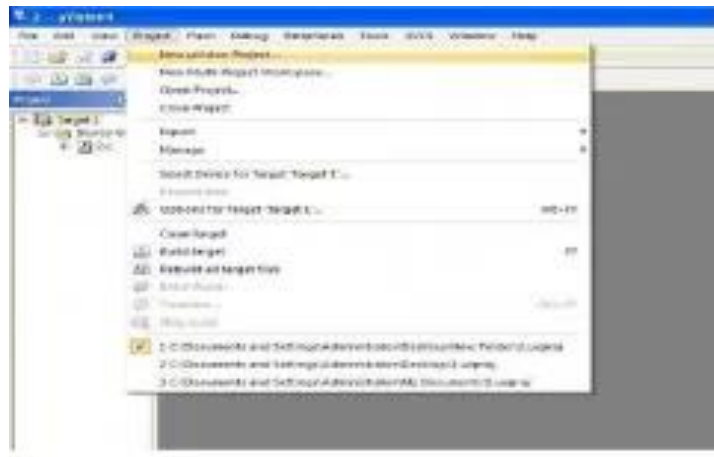
AT89C51 needs an oscillator for its clock generation, so we should connect external oscillator. Two 22pF capacitors are used to stabilize the operation of the Crystal Oscillator. EA should be strapped to VCC for internal program executions. AT89C51 has no internal Power on Reset, so we have to do it externally through the RST pin using Capacitor and Resistor. When the power is switched ON, voltage across capacitor will be zero, thus voltage across resistor will be 5V and reset occurs. As the capacitor charges voltage across the resistor gradually reduces to zero. This pin also receives the 12-volt programming enable voltage (VPP) during Flash programming, for parts that require 12-volt VPP.

Procedure:

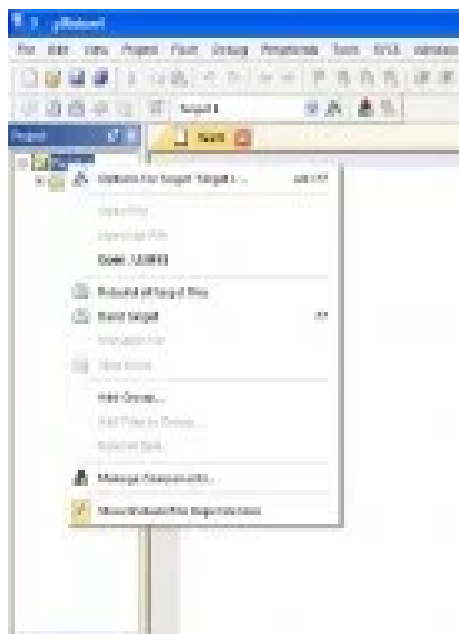
1. Download and Install Keil uVision4
2. Open Keil uVision
- 3.



4. Create a new Project : Project >> Create μ Vision Project



5. Browse for the location
6. Select the microcontroller Atmel>>AT89C51
7. Don't Add The 8051 startup code
8. **File>>New**
9. Adding Hex file to the output



10. Right click on **Target1>>options for target “target 1”** in the Output Tab check the “Create HEX file” box<


```

sbit LED = P2^0;      // Defining LED pin

void Delay(void);      // Function prototype declaration

void main (void)
{
    while(1)           // infinite loop
    {
        LED = 0;       // LED ON
        Delay();
        LED = 1;       // LED OFF
        Delay();
    }
}

void Delay(void)
{
    int j;
    int i;
    for(i=0;i<10;i++)
    {
        for(j=0;j<10000;j++)
        {
        }
    }
}

```

1. Enter the source code.
2. Save it
3. Then Compile it. Click Project>>Build Target or F7

The hex file will be generated in our Project Folder.

Result:

Thus using 8051 microcontroller with Keil, blinking of LED is performed.

Exp. No.: 3 STEPPER MOTORINTERFACING USING 8051 MICROCONTROLLER AND KEIL C- AT89C51

Date:

Aim:

To write a program to perform interfacing of stepper motor using 8051 microcontroller and Keil C-AT89C51.

Components required:

- 8051 Microcontroller
- Keil uVision4

Theory:

Stepper motors are type of DC motors. Stepper motor has multiple electromagnetic coils that are arranged in group called phases. Motor rotates when particular phase is energized. One step rotation occurs at a time by energizing a particular coil. We can also control the speed of motor. It can be interfaced with 8051 microcontroller but the same case is here about DC motors that we can't connect them directly with controller.

Stepper Motor Interfacing with 8051 Microcontroller:

Due to high voltage and current limitations of microcontroller, a motor Driver IC is used. We can use L293D or ULN2003.

Circuit Components:

- AT89C51 microcontroller
- 12 MHz Oscillator.
- 12V DC battery.
- 5V DC battery.
- L293D motor driver
- Unipolar Stepper motor – 1.
- 2 Ceramic capacitors – 33pF
- 300Ω resistors – 3
- Push buttons – 3

Connections:

- P2 (Lower four pins) of 8051 microcontroller is used as output port and it gives inputs to the L293D (motor driver IC) to drive one stepper motor.
- P0 (Lower three pins) of 8051 microcontroller is used as input port. 3 Buttons are connected to them so that we can manually start and stop the motor.
- Stepper motor is given input through OUT1, OUT2, OUT3 and OUT4 of L293D.
- 12V battery is used to give input to the VS for motor.
- 5V battery is used to give input to VSS for motor driver IC.

CODE of stepper motor interfacing with 8051 microcontroller

```
#include<reg52.h>
```

```

void delay(int);
sbit B1 = P0^0;
sbit B2 = P0^1;
sbit B3 = P0^2;
void main()
{
P1=0x07;
P2=0x00;
if(B1==1)                                //for wave drive
{
    P2=0x01;        //0001
    delay(1000);
    P2=0x02;        //0010
    delay(1000);
    P2=0x04;        //0100
    delay(1000);
    P2=0x08;        //1000
    delay(1000);
}
if(B1==0)
{
    P2=0x00000000;
}
if(B2==1)                                //for full drive
{
    P2 = 0x03;        //0011
    delay(1000);
    P2 = 0x06;        //0110
    delay(1000);
    P2 = 0x0C;        //1100
    delay(1000);
    P2 = 0x09;        //1001
    delay(1000);
}
if(B2==0)
{
    P2=0x00000000;
}
if(B3==1)                                //for half drive
{
    P2=0x01;        //0001
    delay(1000);
    P2=0x03;        //0011
    delay(1000);
    P2=0x02;        //0010
    delay(1000);
    P2=0x06;        //0110
    delay(1000);
    P2=0x04; //0100

```

```

    delay(1000);
    P2=0x0C; /    /1100
    delay(1000);
    P2=0x08;    //1000
    delay(1000);
    P2=0x09;    //1001
    delay(1000);
}
if(B3==0)
{
    P2=00000000;
}
}
void delay(int k)
{
    int i,j;
    for(i=0;i<k;i++)
    {
        for(j=0;j<100;j++)
        {}
    }
}
}

```

Result:

Thus using 8051 microcontroller with Keil, interfacing of stepper motor is performed.

Exp. No.: 4 SERVO (DC) MOTOR INTERFACING USING 8051 MICROCONTROLLER AND KEIL C- AT89C51

Date:

Aim:

To write a program to perform interfacing of DC motor using 8051 microcontroller and Keil C-AT89C51.

Components required:

- 8051 Microcontroller
- Keil uVision4

Theory:

Servo motors are used in robotics, embedded systems and industries because they are very precise and reliable. They are used to operate remote control toy cars, airplanes or robots. Their motion can be controlled by rotating them in particular angle. Servo motor can be controlled by PWM signal.

A DC servo motor consists of:

- DC motor
- Potentiometer (variable resistor)
- Gear assembly
- Control circuit

It consists of a closed loop system. Positive feedback is given to control the motion and position of the shaft. Feedback signal is generated by comparing output signal and reference input signal. Now, this feedback signal will act as input signal to control device. This signal will remain present as long as there remains a difference between reference input signal and output signal. So we have to maintain output of this system at desired value in presence of noises.

Code of Servo Motor Interfacing with 8051 Microcontroller:

```
#include <REGX51.H>
void Delay_servo(unsigned int);
sbit control_pin=P2^0;
sbit B1=P0^0;
sbit B2=P0^1;
sbit B3=P0^2;
void main()
{
P0=0x07;           // input port
control_pin=0;      // output pin
do
{
if(B1==1)
{
//Turn to 90 degree
control_pin=1;
Delay_servo(1218);
control_pin=0;
}
else if(B2==1)
```

```

        {
            control_pin=1;
            Delay_servo(1470);
            control_pin=0;
        }
    else if(B3==1)
        {
            control_pin=1;
            Delay_servo(1720);
            control_pin=0;
        }
    }
    while(1);
}

void Delay_servo(unsigned int d)
{
    TMOD &=0xF0;           // Clear 4bit field for Timer0
    TMOD|=0x01;            // Set timer0 in mode1, 16bit
    TH0=0xFF - (d>>8)&0xFF; // Load delay vales Timer 0 + Bitwise right shift[c][d]a >> b
    TL0=0xFF- d&0xFF;
    ET0=1;                 //Enable timer0 interrupts
    EA=0;                  // Global Interrupt
    TR0=1;                 // Start timer 0
    while(TF0==0);         // Wait for overflow
    TR0=0;                 //Stop timer0
    TF0=0;                 // Clear Flag
}

```

Result:

Thus using 8051 microcontroller with Keil, interfacing of servo motor is performed.

Exp. No.: 5 A & B INTERFACING LED & PWM

Date:

Aim

To write a C program for Switch & LED to activate LED's and generate a PWM and to vary the duty cycle .

Pre Lab Questions

1. What happens if the junction temperature of LED is increased?
2. Mention the principle of PWM.
3. What are the materials used to make LED?
4. What are the types of seven segment display.
5. Differentiate LED from LCD.

Apparatus & Software Required:

1. LPC2148 Development board.
2. Keil μ Vision 5 software.
3. Flash Magic.
4. USB cable.
5. CRO

Theory:

The PWM is based on the standard timer block and inherits all of its features, although only the PWM function is pinned out on the LPC2148. The timer is designed to count cycles of the peripheral clock (PCLK) and optionally generate interrupts or perform other actions when specified timer values occur, based on seven match registers. The PWM function is also based on match register events.

Procedure:

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.16

```
*****  
*****/
```

SWITCH AND LED PROGRAM

```
*****  
*****/
```

```
/* Description: This program gets DIP switch inputs and switches ON corresponding LED*/  
/*P1.16 to P1.31 are output switch*/
```

```
*****  
*****/
```

```
#include<LPC214x.H>
```

```
int main()
```

```
{
```

```
IO1DIR=0xFFFF0000;          // P1.16 TO P1.31 OUTPUTPIN
```

```

while(1)
{
IOCLR1 = 0xFFFF0000;    // output pin cleared for enable the led
}
}

```

SWITCH AND LED PORT DETAILS:

ARM	S&L ENABLE PIN
P1.16	S&L ENABLE PIN
P1.17	S&L ENABLE PIN
P1.18	S&L ENABLE PIN
P1.19	S&L ENABLE PIN
P1.20	S&L ENABLE PIN
P1.21	S&L ENABLE PIN
P1.22	S&L ENABLE PIN
P1.23	S&L ENABLE PIN
P1.24	S&L ENABLE PIN
P1.25	S&L ENABLE PIN
P1.26	S&L ENABLE PIN
P1.27	S&L ENABLE PIN
P1.28	S&L ENABLE PIN
P1.29	S&L ENABLE PIN
P1.30	S&L ENABLE PIN
P1.31	S&L ENABLE PIN

```

/*****
PWM.C
*****/
/* Place lcd.c file into following directories C:\Keil\ARM\INC\Philips.*
/* This program is used to Generate the PWM, Frequency and Duty cycle can be changed*/
*****/

```

```

#include<LPC214x.H>
int main(void
{
PINSEL1 |= 0x00000400;    //Enablepin0.7
PWMPR
=0x00000100;
asPWM2                    //Load prescaler (to vary the frequency can modify here)
PWMPCR=0x00002000;        //PWM channel single edge control, output enabled
PWMMCR = 0x00000003;      //On match with timer reset the counter
/* PWMR0 AND PWMR5 Both Value can change the duty cycle ex : PWMR0 = 10 AND PWMR5
= 2*/
PWMMR0 = 0x00000010;      //set cycle rate to sixteen ticks
PWMMR5 = 0x00000008; /    //set rising edge of PWM2 to 2 ticks
PWMLER = 0x00000021;      //enable shadow latch for match 0 - 2
PWMTCR = 0x00000002;      //Reset counter and prescaler
PWMTCR = 0x00000009;      //enable counter and PWM, release counter fromreset
while(1)                  // mainloop
{
}
}

```

PWM PROGRAM PORT DETAIL

ARM	DETAILS
P0.7	PWM2

Post Lab Questions:

- 1.How do the variations in an average value get affected by PWM period?
- 2.Name the common formats available for LED display
- 3.Why are the pulse width modulated outputs required in most of the applications?
- 4.How do you determine the duty cycle of the waveform ?
- 5.What is the function of GPIO?

Result:

The C code is generated for Switch & LED and output is verified in LED's by Switches
The C code is generated for PWM and to vary the duty cycle and verified in CRO output.

Exp. No.: 6 A& B INTERFACING KEYBOARD AND LCD MATRIX KEYBOARD PROGRAM

Date:

Aim

To develop a C-Language program for displaying the Key pressed in the Keypad in the LCD module. The display should come in the desired line and column.

Pre Lab Questions

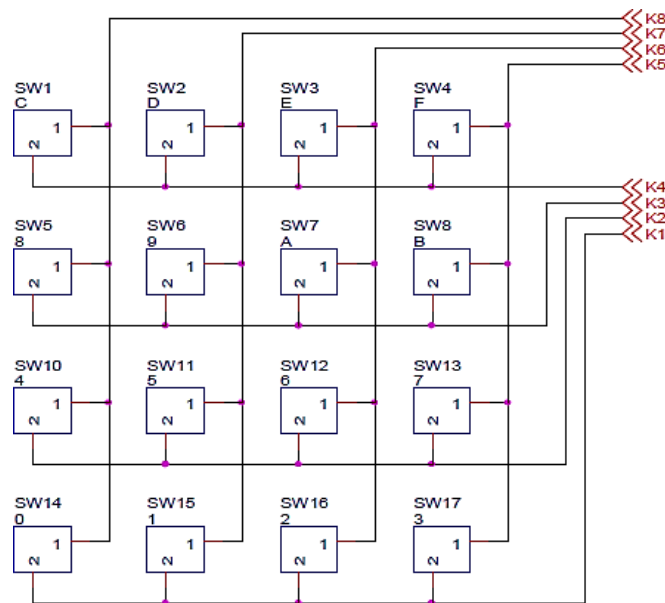
1. Mention the function of pull up resistor?
2. Outline the keyboard matrix.
3. Summarize the working principal of LCD.
4. What kind of interrupt is generated if a key has to be operated in an interrupt mode?
5. How many rows and columns are present in a 16 x 2 alphanumeric LCD?

Apparatus & Software Required

- 1.LPC2148 Development board.
- 2.Keil μ Vision5 software.
- 3.Flash Magic.
- 4.USB cable.

Theory

The Matrix keyboard is used to minimize the number of I/O lines. Normally it is possible to connect only one key or switch with an I/O line. If the number of keys in the system exceeds the more I/O lines are required. To reduce the number of I/O lines the keys are connected in the matrix circuit. Keyboards use a matrix with the rows and columns made up of wires. Each key acts like a switch. When a key is pressed a column wire makes contact with row wire and completes a circuit. For example 16 keys arranged in a matrix circuit uses only 8 I/O lines.



Procedure

1. Follow the steps to create a New project

2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

```

/*****/
MAIN.C
*****/
/* Description: This program gets input from Matrix key board and displays
corresponding */
/* Key value in 7segment display. Hence this program demonstrates both*/
7 segment display as well as Matrix keyboard.*/
/*P1.16 to P1.23 are inputs from matrix keyboard,*/
/*P1.24 to P1.31 are outputs to 7 segmentdisplay
*****/
*****/
/*      ----- matrix key boarddescription----- */
/*      --      --      --      --      */
/* row1    --| c |-- --| d |-- --| e |-- --|F|--      (SW1,SW2,SW3,SW4)      */
/*      --      --      --      --      */
/*      --      --      --      --      */
/* row2    --| 8 |-- --| 9 |-- --| A |-- --|b|--      (SW5,SW6,SW7,SW8)      */
/*      --      --      --      --      */
/*      --      --      --      --      */
/* row3    --| 4 |-- --| 5 |-- --| 6 |-- --|7|--      (SW9,SW10,SW11,SW12) */
/*      --      --      --      --      */
/*      --      --      --      --      */
/* row4    --| 0 |-- --| 1 |-- --| 2 |-- --|3|--      (SW13,SW14,SW15,SW16) */
/*      --      --      --      --      */
*****/
#include <LPC214x.h>
#include "mat_7seg.h"
int main()
{
    unsigned int key, last_key, Disp_key;
    init_Matrix_7seg();           // Initialize matrix keyboard and 7segment display
    clearall_7seg();              // clear 7 segmentdisplay
    last_key=0;                   // Initialize this variable to zero
    while(1)
    {
        key=catch_key();          // scan for a valid keypress
        if(key!=0)                // zero means no key ispressed
        {
            if(key!=last_key)     // check whether the same key is pressed again      (assume this
                                as STEP1)
            Disp_key=key;
            last_key=key;          // valid new key is stored in another variable
                                // this variable's value is used forSTEP1
        }
    }
}

```

```

}
}

//Display_Number(Disp_key);

decimalformat*/
Alpha_Dispay(4,Disp_key);
(single digitonly)*/
}
/*this function is used to display number in
/*this function is used to display number in hex format
}
/*****
MATRIX SEVEN SEGMENT DRIVER.C
*****/
#include <LPC214x.h>
#include "defs.h"
/*****Global
variables*****/
unsigned int thousands,hundreds,tens,ones;
/*****
void init_Matrix_7seg(void)
{
IODIR1 |= 0xff0f0000;    // set 7seg LEDs as output ports and matrix's MSB as
inputs and LSB as outputs
IODIR0|=S7SEG_ENB;
// set P0.19 to P0.22 as outputs to drive 7segenable

pins
IOPIN0|=S7SEG_ENB;
// since we are using active low 7 seg display,the

enable signals
// should be initially set to HIGH.
}
/*****
unsigned long scan_row(unsigned int row_num)
{
//unsigned int row,i;

unsigned long val;
IOSET1=ROW_MASK;    //clear the previous scan row output ie make all row opshigh
switch(row_num)
{
case 1: IOCLR1 = ROW1;break;    // make P1.16 low
case 2: IOCLR1 = ROW2;break;    // make P1.17 low
case 3: IOCLR1 = ROW3;break;    // make P1.18 low
case 4: IOCLR1 = ROW4;break;    // make P1.19 low
//default: row = ERR;

}
for(i=0;i<=65000;i++);
val=IOPIN1;

// read the matrixinputs
val = ((val >> 20) & 0x0000000F)^0x0000000F; // shift the colum value so that it comes to LSB

```

```

//
// XORing is done to take 1's complement of shifted value.//
return(val);
}
unsigned int catch_key(void)
{
    unsigned long v; v =
    scan_row(1);
    switch(v)
    {
        case 1: return(13);
        case 2: return(14);
        case 4: return(15);
        case 8: return(16);
    }
    v = scan_row(2);
    switch(v)
    {
        case 1: return(9);
        case 2: return(10);
        case 4: return(11);
        case 8: return(12);
    }
    v = scan_row(3);
    switch(v)
    {
        case 1: return(5);
        case 2: return(6);
        case 4: return(7);
        case 8: return(8);
    }
    v = scan_row(4);
    switch(v)
    {
        case 1: return(1);
        case 2: return(2);
        case 4: return(3);
        case 8: return(4); default:
        return(0);
    }
}
/*****/
void clearall_7seg(void)
{
    IOPIN1 &= ~S7SEG_LED;          // make all the 7seg led pins toLOW
    IOPIN0|=S7SEG_ENB
                                   // Disable all the 7 segdisplay
}
/*****/

```

```

void clearDigit_7seg(int digit_num)
{
IOPIN0 |= S7SEG_ENB;           // clear enables first
switch(digit_num)
{
case 1: {
IOPIN0 = ~DIGI1_ENB;           // now enable only the digit1
break;
}
case 2: {
IOPIN0 = ~DIGI2_ENB;
// now enable only the digit2
break;
}
case 3: {
IOPIN0 = ~DIGI3_ENB;
// now enable only the digit3
break;
}
case 4: {
IOPIN0 = ~DIGI4_ENB;
}
}
IOPIN1 &= ~S7SEG_LED;           // now enable only the digit4 break;
// make all the 7seg LED pins LOW
}
/*****/
void Digit_Display(int digit_num, unsigned int value)
{
clearDigit_7seg(digit_num);
switch(value)
{
case 0: IOPIN1 |= ZERO; break;
case 1: IOPIN1 |= ONE; break;
case 2: IOPIN1 |= TWO; break;
case 3: IOPIN1 |= THREE; break;
case 4: IOPIN1 |= FOUR; break;
case 5: IOPIN1 |= FIVE; break;
case 6: IOPIN1 |= SIX; break;
case 7: IOPIN1 |= SEVEN; break;
case 8: IOPIN1 |= EIGHT; break;
case 9: IOPIN1 |= NINE; break;
}
}
/*****/
void Alpha_Display(int digit_num, unsigned int value)
{
clearDigit_7seg(digit_num);
switch(value)
{
case 1: IOPIN1 |= ZERO; break;

```



```

case 2: IOPIN1 |= ONE; break;
case 3: IOPIN1 |= TWO; break;
case 4: IOPIN1 |= THREE; break;
case 5: IOPIN1 |= FOUR; break;
case 6: IOPIN1 |= FIVE; break;
case 7: IOPIN1 |= SIX; break;
case 8: IOPIN1 |= SEVEN; break;
case 9: IOPIN1 |= EIGHT; break;
case10:
IOPIN1 |= NINE; break; case
11: IOPIN1 |= AAA; break; case 12:
IOPIN1 |= bbb; break; case 13: IOPIN1 |=
ccc; break; case 14: IOPIN1 |= ddd; break;
case 15: IOPIN1 |= eee; break; case 16:
IOPIN1 |= fff;break;
}
}
void split_numbers(unsigned int number)
{
thousands = (number /1000);
number %= 1000;
hundreds = (number / 100);
number %= 100;
tens = (number / 10);
number %= 10;30
ones = number ;
}
void Display_Number(unsigned int num)
{
unsigned int i;
if(num <= 9999)
{
clearall_7seg();
split_numbers((unsignedint)num);
Digit_Dispay(4, ones);
for(i=0;i<10000;i++);
Digit_Dispay(3, tens);
for(i=0;i<10000;i++);
Digit_Dispay(2,hundreds);
for(i=0;i<10000;i++);
Digit_Dispay(1,thousands);
for(i=0;i<10000;i++);
}
}

```

MATRIX SEVEN SEGMENT PROGRAM PORT DETAILS

ARM	DETAILS
P0.19	SEGMENT ENABLE PIN
P0.21	SEGMENT ENABLE PIN
P0.22	SEGMENT ENABLE PIN
P1.16	KEY BOARD INPUT
P1.17	KEY BOARD INPUT
P1.18	KEY BOARD INPUT
P1.19	KEY BOARD INPUT
P1.20	KEY BOARD INPUT
P1.21	KEY BOARD INPUT
P1.22	KEY BOARD INPUT
P1.23	KEY BOARD INPUT
P1.24	OUTPUT SEGMENT
P1.25	OUTPUT SEGMENT
P1.26	OUTPUT SEGMENT
P1.27	OUTPUT SEGMENT
P1.28	OUTPUT SEGMENT
P1.29	OUTPUT SEGMENT
P1.30	OUTPUT SEGMENT

LCD PROGRAM

/******

LCD.h

/******

void clrscr(char ch);

void lcdinit(void);

void lcdcmd(char);

void lcdat(char);

void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or1

void printstr(char*,char,char);

//string,column(x),line(y)

void wait (void);

void split_numbers(unsigned int number);

```

void Wait_Msg(void);
void Welcome_Msg(void);
/*****
LCD.c
*****/
#include <LPC214x.h>
#define RS
#define CE
0x00000400
0x00001800
/* P0.10 */
/* P1.11 */
void clrscr(char ch); void
lcdinit(void); void
lcdcmd(char); void
lcdat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or 1
void printstr(char*,char,char);
//string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number);
#define SET1
#define OFF0
unsigned int thousands,hundreds,tens,ones;
void wait(void)
{
d;
for (d = 0; d <100000;d++);
}
void lcdinit()
{
IODIR0 |= 0x0000FFFF;
IOCLR0 |= 0x00000000;
lcdcmd(0x28);lcd
cmd(0x28);
lcdcmd(0x0c);
lcdcmd(0x06);
lcdcmd(0x01);
/* wait function */ int
/* only to delay for LED flashes*/32
lcdcmd(0x0f);wait();
}
void gotoxy(char x, char y)
{
if(y == 0)
lcdcmd(0x80+x);
else
lcdcmd(0xc0+x);
}

```

```

void printstr(char *str, char x, char y)
{
char i; gotoxy(x,y);
wait();//(500);
for(i=0;str[i]!='\0';i++)lcddat(str[i
]);
}
void lcdcmd(char cmd)
{
unsigned charLCDDAT;
LCDDAT = (cmd&0xf0);
IOSET0 =LCDDAT;
IOCLR0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
LCDDAT = ((cmd<<0x04)&0xf0);
IOSET0 =LCDDAT;
IOCLR0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
//higher nibble
//enablelcd
//lower nibble
//enablelcd
}
void lcddat(char cmd)
{
unsigned charLCDDAT;
LCDDAT = (cmd&0xf0);
IOSET0 =LCDDAT;
IOSET0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
LCDDAT = ((cmd<<0x04)&0xf0);
IOSET0 =LCDDAT;
IOSET0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
//higher nibble
//enablelcd
//lower nibble
//enablelcd33

```

```

IOCLR0 = 0X00000FFF;
}
void clrscr(char ch)
{
if(ch==0)
{
printstr("
gotoxy(0,0);
}
else if(ch ==1)
{
printstr("
gotoxy(0,1);
}
else
{
lcdcmd(0x01);
//delay(100);
}
}
void split_numbers(unsigned int number)
{
thousands = (number /1000);
number %= 1000;
hundreds = (number / 100);
number %= 100;
tens = (number / 10);
number %= 10;
ones = number ;
}
void Wait_Msg(void)
{
lcdcmd(0x01);
printstr("
PLEASEWAIT ", 0,0);
}
void Welcome_Msg(void)
{
lcdcmd(0x01);
printstr("
WELCOMETO ", 0,0);
printstr("SM MICRRO SYSTEM", 0,1);
}
",0,0);
",0,1);34
/*****/

LCDmain.c
/*****/

```

```

/* This is a test program to display strings in LCD module in theARMLPC2148Development board
   itself*/
/*****
#include<LPC214x.H>
#include"lcd.h"
int main (void)
{
lcdinit();
Wait_Msg();
while(1)
{
/* LPC214x definitions*/
/* includes lcd driverfuntions*/
/*Initializelcd*/
/*Display message - "Please Wait"*/ Welcome_Msg();
/*Display message - "Welcome to SMMICRRO"*/
/*LoopForever*/
}

```

LCD PROGRAM PORT DETAILS:

ARM	Details
PO.10	RS LCD PIN
P1.11	CE LCD PIN

Post Lab Questions

1. Outline the operations involved when the key in a 4 x 4 keyboard matrix is beingpressed.
2. List the registers used to store the keyboard, display modes and other operations programmed by CPU.
3. What is switch bouncing ? How to prevent it using de-bounce circuit?
4. How to adjust the contrast of the LCD?
5. Which command of an LCD is used to shift the entire display to the right?

Result

The C-Language program for displaying the Key pressed in the Keyboard is displayed in the seven segment display and LCD module and the output was verified on the LCD on the desires line and column/address.

Exp. No.: 7A&B

INTERFACING ADC & DAC

Date:

Aim

To develop a C-Language program for reading an on-chip ADC, convert into decimal and to display it in PC and to generate a square wave depending on this ADC reading. The ADC input is connected to any analog sensor/ on board potentiometer.

Pre Lab Questions

1. List the types of ADC and DAC
2. Define resolution.
3. Summarize the features of Conversion time in ADC.
4. What is the function of Sample-and-hold circuits in analog-to digital converters?
5. Why are internal ADCs preferred over external ADCs?

Apparatus & Software Required

- 1.LPC2148 Development board.
- 2.Keil μ Vision5 software.
- 3.Flash Magic.
- 4.USB cable.
- 5.CRO

Theory

The LPC 2148 has 10-bit successive approximation analog to digital converter. Basic clocking for the A/D converters is provided by the VPB clock. A programmable divider is included in each converter, to scale this clock to the 4.5 MHz (max) clock needed by the successive approximation process. A fully accurate conversion requires 11 of these clocks. The ADC cell can measure the voltage on any of the ADC input signals.

ARM Board has one potentiometer for working with A/D Converter. Potentiometer outputs are in the range of 0V to 3.3V. Switch select in right position for reading the Potentiometer value by ADC.8.

Procedure

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.9

MAIN ADC TEST

```
/* **** */
/* This is a test program to ADC in theARMLPC2148      developmentboard*/
/* **** */
#include<LPC214x.H>
#include"ADC_Driver.c"
#include <stdio.h>
/* LPC214x definitions*/
/* contains prototypes of driverfunctions*/
```

```

#include"lcd.c"
int main (void)
{
    unsigned int adc_val;
    unsigned int temp;
    unsigned char buf[4] ={0,0,0,0}; ADCInit();
    lcdinit();
    //wait();
    clrscr(10);
    printstr("ADC Test",0,0); wait();
    while(1)
    /* Loop forever*/
    {
        adc_val = ADC_ReadChannel();
        temp = (unsigned int)((3*adc_val*100)/1024);
        sprintf(buf,"%d",temp);
        printstr(buf,0,1);
    }
}
/*****/
LCD.C
/*****/
#include <LPC214x.h>
#define RS
0x00000400 /* P0.10 */
#define CE
0x00001800 /* P1.11*/
void clrscr(char ch); void
lcdinit(void); void
lcdcmd(char); void
lcdat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or1 void
printstr(unsignedchar*,char,char);
//string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number);
#define SET1
#define OFF0
unsigned int thousands,hundreds,tens,ones;10
voidwait(void)
{
    /* wait function */
    int d;
    for (d = 0; d <100000;d++);
}
/* only to delay for LED flashes*/
void lcdinit()
{
    IODIR0 |= 0xFFFFFFF;

```



```

IOCLR0 |= 0X00000FFF;
lcdcmd(0x28);lcd
cmd(0x28);
lcdcmd(0x0c);
lcdcmd(0x06);
lcdcmd(0x01);
lcdcmd(0x0f);
wait();
}
void gotoxy(char x, char y)
{
if(y == 0)
lcdcmd(0x80+x);
else
lcdcmd(0xc0+x);
}
void printstr(unsigned char *str, char x, char y)
{
char i; gotoxy(x,y);
wait();//(500);
for(i=0;str[i]!='\0';i++)lcddat(str[i
]);
}
void lcdcmd(charcmd)
{
unsigned charLCDDAT;
LCDDAT = (cmd&0xf0);
IOSET0 =LCDDAT;
IOCLR0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
LCDDAT = ((cmd<<0x04)&0xf0);
IOSET0 =LCDDAT;
IOCLR0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
}
void lcddat(char cmd)
//higher nibble
//enablelcd
//lower nibble
//enablelcd11
{
unsigned charLCDDAT;
LCDDAT = (cmd&0xf0);

```

```

IOSET0 =LCDDAT;
IOSET0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
//higher nibble
//enablelcd
LCDDAT = ((cmd<<0x04)&0xf0);
IOSET0 =LCDDAT;
IOSET0 = RS;
IOSET0 =CE;
wait();//(100);
IOCLR0 =CE;
IOCLR0 = 0X00000FFF;
//lower nibble
//enable lcd
}
void clrscr(char ch)
{
if(ch==0)
{
printstr("
gotoxy(0,0);
}
else if(ch == 1)
{
printstr("
gotoxy(0,1);
}
else
{
lcdcmd(0x01);
//delay(100);
}
}
void split_numbers(unsigned int number)
{
thousands = (number /1000);
number %= 1000;
hundreds = (number / 100);
number %= 100;
tens = (number / 10);
number %= 10;
ones = number ;
}
void Wait_Msg(void)
{
lcdcmd(0x01);

```

```

    printstr("
Please Wait ", 0,0);
}
void Welcome_Msg(void)
{
    lcdcmd(0x01);
    printstr("
Welcometo
", 0,0);
    printstr("
SMMICRRO ", 0,1);
}
",0,0);
",0,1);12
/*****/
ADC_ DRIVER.C
/*****/
#include<LPC214x.H>
/* LPC214x definitions */
Void ADCInit(void)
{
    PINSEL1|=0x04000000;
    IODIR0 |=~(0x04000000);
    AD0CR |=0x00200204;
    AD0GDR;
}
/*For Channel AD0.2 is P0.29*/
/*0x04 selects AD0.2 to mux output, 0x20 makes ADCin operational*/
/*A read on AD0GDR clears the DONEbit*/
void ADC_StartConversion(void)
{
    AD0CR |= (1<<24);
}
void ADC_StopConversion(void)
{
    AD0CR &= (~ (1<<24));
}
unsigned int ADC_ReadChannel(void)
{
    //unsigned int i; unsigned long
    ADC_Val, t;
    ADC_StartConversion();
    while((AD0DR2&0x80000000)==0); /*wait until ADC conversion completes*/
    if(AD0STAT & 0x00000400)
    {
        //printstr("OVR",0,1);
    }
    return(0);
}
t = AD0DR2;
ADC_Val = ((t>>6) & 0x000003FF);    //(AD0DR2 & 0x000003FF); (((AD0CR>>6) &
0x000003FF);

```

```
//ADC_StopConversion();
return(ADC_Val);
```

ADC PROGRAM PORT DETAILS:

ARM

P0.29

PO.10

P1.11

DETAILS

ADC0.2

RS LCD PIN

CE LCD PIN

DAC PROGRAM

```
/****** / DAC.C
```

```
***** /
```

This is a test program to DAC in the ARM LPC2148 Development board

```
***** /
```

```
#include<LPC214X.H>
```

```
void wait_long (void)
```

```
{
```

```
/* wait function*/
```

```
int
```

```
d;
```

```
for (d = 0; d <1000000;d++);
```

```
}
```

```
/* only to delay*/
```

```
int main()
```

```
{
```

```
wait_long();wai
```

```
t_long();
```

```
IODIR0 = 0X00000FFF;
```

```
IODIR1 = 0XFFFF0000;
```

```
IOSET0 = 0XFFFFFFFF;
```

```
IOCLR1 = 0XFFFF0000;
```

```
PINSEL1 |= 0x00080000; //Enablepin0.25
```

```
asDAC
```

```
DACR=0X00017FC0; // 000 = 0V (min),7FC = 1.6V,7FF =3.3V(max)
```

```
While (1);
```

```
}
```

DAC PROGRAM PORT DETAILS:

ARM

P0.25

DETAILS

DAC ENABLE PIN

Post Lab Questions:

1. What are the ADC operating modes in LPC2148?
2. What is the function of A/D Status Register
3. Which pin provides a voltage reference level for the D/A converter?
4. What is Burst conversion mode?
5. What is settling time?

Result :

The C-Language program for reading an on-chip ADC, convert into decimal and to display it in PC was written & output is verified with the ADC input is connected to on board potentiometer

The DAC, convert digital data into analog signal& output is verified with the DAC input and the square wave has been generated to display it in CRO.

INTERFACING REAL TIME CLOCK PROGRAM AND SERIAL PORT

1. Follow the steps to create a New project

2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash

Magic Software.20

```
/******RTC.C*****/  
/* Place lcd.c file into following directories C:\Keil\ARM\INC\Philips.*****/  
/* This program is used to interface the RTC.You can change the date and time*/  
/* If you want. This Program can both Read and write data into RTC.RTC has a*/  
/* Battery backup for continuous Running. *****/  
/*  
pclk = 30,000,000 Hz  
PREINT = (int)(pclk/32768)-1  
PREFRAC = pclk - ((PREINT+1) x 32768)  
*/  
#include<LPC214X.H>  
#include<lcd.c>  
int main()  
{  
    unsigned int hrs,min,sec;  
    wait();  
    wait();  
    wait();  
    wait();  
    lcdinit();clrscr  
    cr(2);  
    printstr("SM MICRROSYSTEM",0,0);  
    printstr("ARMDEVKIT ",0,1);  
    VPBDIV = 0x00000002; // VPB bus clock is one half of the processor clock(cclk)  
    PREINT = 0x00000392; // Set RTC prescaler for 30MHz Pclk  
    // PREINT = (int) (30,000,000/32768(RTC  
    crystal))-1 = 914  
    PREFRAC = 0x00004380;  
    CIIR=0x00000001;  
    // Enable seconds counterinterrupt  
    CCR=0x00000001;  
    // Start theRTC  
    YEAR=2009;  
    // Year  
    MONTH=11;  
    //Month  
    DOM=25;  
    // Day of month  
    DOY=0;  
    // Day ofyear  
    DOW=0;  
    // Day of week  
    HOUR=18;  
    //Hours
```

```

MIN=30;
// Minutes
SEC =30;
printstr("
",0,1);
while(1)
{
gotoxy(0,1);hrs
= HOUR; min
= MIN; sec =
SEC;
split_numbers(hrs);lcdd
at(tens+0x30);
lcddat(ones+0x30);
lcddat(':');
split_numbers(min);lcd
dat(tens+0x30);21
lcddat(ones+0x30);
lcddat(':');
split_numbers(sec);lcdd
at(tens+0x30);
lcddat(ones+0x30);
//lcddat(':');
}
}

```

RTC PROGRAM PORT DETAILS

PORT INBUILT

SERIAL PORT PROGRAM

```

/*****
/* Uart0 Initialization */
/* This is a test program to send and receive data via uart0 in theARMLPC2148
Development board itself
*****/
#include<LPC214x.H>
#include"uart0.h"
/* LPC214x definitions*/
/* contains prototypes of driverfunctions*/
int main (void)
{
unsigned char *s1;
initserial();
/* uart0 initialization */
send_string("*****");
send_string("SMMicroSystem");
send_string("Tambaram");
send_string("Chennai");

```



```

send_string("*****");
send_string("");send_string("");
send_string("This program Echos the string entered byuser."); send_string("So,type
some strings and press ENTERkey");
while(1)
{
s1 = receive_string();
send_string(s1);
}
/* Loop forever*/
}
/*****/

SERIAL PORT PROGRAM
/*****/
/* Uart1 Initialization */
/*****/

#include<lpc214x.h>#inc
lude<stdio.h>
#include<stdlib.h>
#include "uart1_driver.c"
int main()
{
unsigned char *a;
//unsigned char *w;
a=malloc(sizeof(100));
inituart1();
sendstring1("ABCDEFGHJKLMNOPQRSTUVWXYZ");
sendstring1("ABCDEFGHJKLMNOPQRSTUVWXYZ");
*/23
sendstring1("ABCDEFGHJKLMNOPQRSTUVWXYZ");
sendstring1("ABCDEFGHJKLMNOPQRSTUVWXYZ");
sendstring1("ABCDEFGHJKLMNOPQRSTUVWXYZ");
/*
sendstring1("");
sendstring1("** ");
sendstring1("*** "); s
endstring1("** *** "); sendstring1(" * * * * ");
sendstring1(" SM MICRRO ");
sendstring1(" * * * * ");
sendstring1("** *** "); sendstring1("*** ");
sendstring1("** ");
sendstring1("");
*/
while(1)
{
receivestring1(a);send
string1(a);
}
}

```

SERIAL PROGRAM PORT DETAILS UART0

ARM

P0.0

P0.1

DETAILS

TXDO

RXDO

UART1

ARM

P0.8

P0.9

DETAILS

TXD1

RXD1

Post Lab Questions:

1. What is I2C and how does it work?
2. Summarize the features of I2C in LPC2148 ARM7 microcontroller.
3. Through which port the date and time is displayed in RTC?
4. What is a serial port?
5. List the registers used to transfer data in serial port.

Result

The C-Language program for reading RTC and displaying it in LCD was written & output is verified with running the RTC from a default/specified time.

ExpNo:9A&B

INTERFACING EPROM AND INTERRUPT

Date:

Aim

To develop a C-Language program to write and read a data in EEPROM and also to analyze its performance with the interrupt.

Pre Lab Questions:

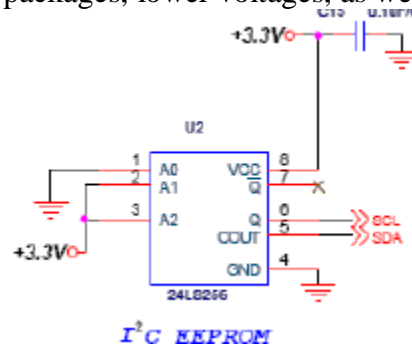
1. What is an edge triggering ?
2. Mention the advantages and disadvantages of level triggering pulse.
3. Differentiate EPROM and ROM.
4. List the different types of Memory devices.
5. Which interrupt is said to be non maskable interrupt , Why?

Apparatus & Software Required:

1. LPC2148 Development board.
2. Keil μ Vision5 software.
3. Flash Magic.
4. USB cable.

Theory:

Serial-interface EEPROM's are used in a broad spectrum of consumer, automotive, telecommunication, medical, industrial and PC related markets. Primarily used to store personal preference data and configuration/setup data, Serial EEPROM's are the most flexible type of nonvolatile memory utilized today. Compared to other NVM solutions, Serial EEPROM devices offer a lower pin count, smaller packages, lower voltages, as well as lower powerconsumption.



Procedure:

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

EPROM PROGRAM

```
/******  
I2C .C  
/******  
/* This Program For I2C Interface */  
#include<LPC214x.H>  
#include "lcd.c"  
void InitI2C (void);
```

```

void SendI2C Address(unsigned char Addr_S); void
WriteI2C (unsigned char Data);
void StopI2C (void); void
StartI2C (void);
#define STA 0x20
#define SIC 0x08
#define
SI
0x08
#define STO 0x10
#define STAC 0x20
#define AA 0x04
void InitI2C (void)
{
I2C 0CONCLR = 0xFF;
PINSEL0 |= 0x50;
I2C 0SCLL = 19;
I2C 0SCLH = 19;
I2C 0CONSET = 0x40;
}
//Set pinouts as scl and sda
//speed at 100Khz for a VPBClockDivider
//Active Master Mode on I2C bus
void SendI2C Address(unsigned char Addr_S)
{
while(I2C 0STAT != 0x08);
// Wait for start to be completed
I2C 0DAT
= Addr_S;
// Charge slaveAddress
I2C 0CONCLR = SIC | STAC;
// Clear I2C interrupt bit to send the data
while(!(I2C 0CONSET & SI));
// wait till status available
}
unsigned char ReadI2C (void)
{
unsigned char r;
= 4 at 12 MHz37
I2C 0CONCLR = SIC;
I2C 0CONSET = 0x04;
while(!(I2C 0CONSET & 0x8));
0STAT;
wait();
if (r == 0x50){
lcdcmd(0x01);
printf("Read Success", 0, 0);
}
// clear SIC;

```

```

// wait till statusavailable r=I2C
// check for error
// look for "Data byte has been received; ACK has been returned"
return I2C 0DAT;
}
void WriteI2C (unsigned char Data)
{
unsigned char r;
I2C 0DAT
=Data;
// ChargeData
I2C 0CONCLR=0x8;
// SIC; Clear I2C interrupt bit to send the data while(!(I2C
0CONSET&0x8));
// wait till statusavailable
r=I2C 0STAT;
if (r == 0x28)
{
// look for "Data byte in S1DAT has been transmitted; ACK has been received"
lcdcmd(0x01);
printf("Write Success",0,0);
}
}
void StopI2C (void)
{
I2C 0CONCLR = SIC;
I2C 0CONSET =
STO;
while((I2C 0CONSET&STO));
}
void StartI2C (void)
{
I2C 0CONCLR=0xFF;
I2C 0CONSET=0x40;
0CONSET=0x00000020;
// wait for Stopped busI2C
// clear I2C - included if User forgot to "StopI2C ()"
// else this function would hang.
// Active Master Mode on I2C bus I2C
// Start condition
}
int main()
{
unsigned char r;
wait();
wait();
wait();
wait();
lcdinit();clrsc

```

```

r(2);
printstr("SM MICRRO SYSTEM",0,0);
printstr("
ARMDEVKIT ",0,1);
InitI2C ();
StartI2C ();
SendI2C Address(0xa0);
// EEPROM device address
WriteI2C (0);
// Set the control portvalue
WriteI2C ('B');38
StopI2C
();wait();
wait();
StartI2C ();
SendI2C Address(0xa0);
// EEPROM device address
WriteI2C (0);
// Set the control portvalue
StopI2C ();
StartI2C ();
SendI2C Address(0xa1);
// Start the read
r=ReadI2C ();
// read the result
StopI2C ();
gotoxy(0,1);
split_numbers(r);
lcddat(0x30+hundreds);lcdda
t(0x30+tens);
lcddat(0x30+ones); while(1);
}
/*****/
LCD.C
/*****/
#defineRS
0x00000400 /* P0.10 */
#defineCE
0x00001800 /* P1.11 */
void clrscr(char ch); void
lcdinit(void); void
lcdcmd(char); void
lcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or1
voidprintstr(char*,char,char);
//string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number);
#define SET1

```

```

#define OFF0
unsigned int thousands,hundreds,tens,ones;
void wait(void)
{
/* wait function */
int d;
for (d = 0; d < 100000; d++);
/* only to delay for LED flashes*/
}
void lcdinit()
{
IODIR0 = 0xFFFFFFF;
IOCLR0 = 0X00000FFF;
lcdcmd(0x28);
lcdcmd(0x28);
lcdcmd(0x0c);
lcdcmd(0x06);
lcdcmd(0x01);
lcdcmd(0x0f);
wait();//(1600);
}
void gotoxy(char x, char y)
{
if(y == 0)
lcdcmd(0x80+x);
else
lcdcmd(0xc0+x);
}
void printstr(char *str, char x, char y)
{
char i; gotoxy(x,y);
wait();//(500);
for(i=0;str[i]!='\0';i++)lcddat(str[i]);
}
void lcdcmd(char cmd)
{
unsigned char LCDDAT;
LCDDAT = (cmd & 0xf0);
IOSET0 = LCDDAT;
IOCLR0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
LCDDAT = ((cmd << 0x04) & 0xf0);
IOSET0 = LCDDAT;
IOCLR0 = RS;
IOSET0 = CE;
}

```

```

wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
//higher nibble
//enablelcd
//lower nibble
//enablelcd
}
void lcddat(char cmd)
{
unsigned charLCDDAT;
LCDDAT = (cmd&0xf0);
IOSET0 =LCDDAT;
IOSET0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
LCDDAT = ((cmd<<0x04)&0xf0);
IOSET0 =LCDDAT;
IOSET0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
}
void clrscr(char ch)
{
//higher nibble
//enablelcd
//lower nibble
//enablelcd40
if(ch==0)
{
printstr("
gotoxy(0,0);
}
else if(ch ==1)
{
printstr("
gotoxy(0,1);
}
else
{
lcdcmd(0x01);
//delay(100);
}
",0,0);
",0,1);

```



```

}
void split_numbers(unsigned int number)
{
thousands = (number /1000);
number %= 1000;
hundreds = (number / 100);
number %= 100;
tens = (number / 10);
number %= 10;
ones = number ;
}
EPROM (I2C )

```

PROGRAM PORT DETAILS

ARM	DETAILS
PO.10	RS LCD PIN
P1.11	CE LCD PIN
P0.11	SCL
P0.14	SDA41

INTERRUPT BUZZER PROGRAM

```

/*****
ExtDriver.C
*****/
#include <LPC214x.h>
void init_VIC(void)
{
/* initialize VIC*/
VICIntEnClr
=0xffffffff;
VICVectAddr =0;
VICIntSelect =0;
}
void ExtInt_ISR(void)irq
{
//EXTINT=(1<<2);
/* clear EINT2 flag by writing HIGH to corespondingbit*/
//IOCLR0 = 0x40000000; /* Trigger the relay*/
IOCLR1 = 0x400f0000; /* P1.18 Trigger the relay*/
//IOPIN1 = 0x00000000;
EXTINT = (1<<2);
VICVectAddr=0;
/* Acknowledge Interrupt*/
}
void init_Interrupt(void)
{
PINSEL0=0x80000000;
// select P0.15 for EINT2
VICIntEnable = (1<<16);

```

```

// External interrupt 2(EINT2)
VICVectCntl0=(1<<5)|(16);
// set the VIC control reg for EINT2
VICVectAddr0 = (unsignedlong)ExtInt_ISR;
EXTMODE&=~(1<<2);
// set VIC for edge sensitive forEINT2
//
EXTPOLAR = ~(1<<2); // set VIC for falling edge sensitive forEINT2
}
void init_ports(void)
{
IODIR0 = 0x40000000;
IODIR1 = 0x400f0000;
IOPIN1 = 0xff010000;
IOSET0 = 0x40000000;
IOSET1 = 0x400f0000;
}
/*void wait_for_turnoffRelay(void)
{
int val;
val=IOPIN1;
while((~(val>>20))!=0);
// is pressed
IOCLR0=0x00010000;
}*/
// read the ports for key board input
// wait until 1st key in the matrixkeyboard
// switch off the relay42
/*****/
XINTR_RELAY.C
/*****/
#include <LPC214x.h>
#include "ext.h"
int main()
{
init_VIC();
init_Interrupt();init_ports();
while(1)
{
//wait_for_turnoffRelay();
}
}

```

INTERRUPT BUZZERPROGRAM

ARM

P1.18
P0.15

DETAILS

TRIGGER THE RELAY
EINT2

Post Lab Questions:

- 1.What will be the initial values in all the cells of an EPROM ?
- 2.What are the contents of the IE register, when the interrupt of the memory location 0x00 is caused?
- 3.Why normally LJMP instructions are the topmost lines of the ISR?
- 4.Enumerate the features of nested interrupt.
- 5.Illustrate the Master Slave mode.

Result

The C-Language program to write and read a data in EEPROM and also to analyze its performance with the interrupt is developed and is verified.

Exp.No:10

MAILBOX

Date:

Aim:

To develop a 'C' code to create a mailbox and to understand the RTOS functions.

Pre Lab Questions:

1. How does mailbox works in RTOS?
2. What is Semaphore?
3. Differentiate mailbox and queue.
4. List the synchronous and asynchronous modes are there in serial port?
5. Interpret the inter process communication

Apparatus & Software Required:

- 1.LPC2148 Development board.
- 2.Keil μ Vision5 software.
- 3.Flash Magic.
- 4.USB cable.

.

Theory:

Real-time and embedded systems operate in constrained environments in which computer memory and processing power are limited. They often need to provide their services within strict time deadlines to their users and to the surrounding world. It is these memory, speed and timing constraints that dictate the use of real-time operating systems in embedded software.

The "kernel" of a real-time operating system ("RTOS") provides an "abstraction layer" that hides from application software the hardware details of the processor (or set of processors) upon which the application software will run.

In providing this "abstraction layer" the RTOS kernel supplies five main categories of basic services to application software⁴⁴

The most basic category of kernel services is Task Management. This set of services allows application software developers to design their software as a number of separate "chunks" of software -- each handling a distinct topic, a distinct goal, and perhaps its own real-time deadline. Each separate "chunk" of software is called a "task." The main RTOS service in this category is the scheduling of tasks as the embedded system is in operation.

The second category of kernel services is Inter task Communication and Synchronization. These services make it possible for tasks to pass information from one to another, without danger of that information ever being damaged. They also make it possible for tasks to coordinate, so that they can productively cooperate with one another. Without the help of these RTOS services, tasks might well communicate corrupted information or otherwise interfere with each other. Since many embedded systems have stringent timing requirements, most RTOS kernels also provide some basic Timer services, such as task delays and time-outs.

Many (but not all) RTOS kernels provide Dynamic Memory Allocation services. This category of services allows tasks to "borrow" chunks of RAM memory for temporary use in application software. Often these chunks of memory are then passed from task to task, as a means of quickly communicating large amounts of data between tasks. Some very small RTOS kernels that are intended for tightly memory-limited environments, do not offer Dynamic memory allocation.⁴⁵

Many (but not all) RTOS kernels also provide a "Device I/O Supervisor" category of services. These services, if available, provide a uniform framework for organizing and accessing the

many hardware device drivers that are typical of an embedded system.

Procedure:

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash

Magic Software.46

```

/*****
MAILBOX.C
*****/
#include <string.h>
#include <stdio.h>
#include <RTL.h>
#include <LPC214x.H>
#include "config.h"
#include "uart.h" #include
"lcd.h"
OS_TID tsk1;
OS_TID tsk2;
typedef struct{
char msgBuf[MBOX_MSG_BUF_SIZE];
} T_MEAS;
/* LPC214x definitions
/* assigned identification for task 1
/* assigned identification for task 2
/* Message object structure
*/
*/
*/
*/
os_mbx_declare(MsgBox,MAILBOX_MEMORY_POOL_CNT); /* Declare an RTX mailbox*/
_declare_box (mpool,sizeof(T_MEAS),MAILBOX_MEMORY_POOL_CNT);/* Dynamic
memory pool*/
task void send_task (void);
task void rec_task (void);
void main_menu()
{
send_string(USE_UART,"\n\r\n*****");
send_string(USE_UART,"\n\r
SM Micro System,Tambaram,Chennai
");
send_string(USE_UART,"\n\rMailBoxMessageSimulation");
send_string(USE_UART,"\n\r
MAIN MENU");
send_string(USE_UART,"\n\r*****");
send_string(USE_UART,"\n\r\n");
send_string(USE_UART,"\n\rThis program simulates MailBox IPC mechanism.");

```

```

send_string(USE_UART, "\n\rPlease Follow Below Commands"); send_string(USE_UART, "\n\r
- Type any
string and press enter to send"); send_string(USE_UART, "\n\r
the
stringusingmailbox.
"); send_string(USE_UART, "\n\r-PRESS SPACE Key tocheck available MailboxCount");
send_string(USE_UART, "\n\r-PRESS
ESC Key to reset
the input string"); send_string(USE_UART, "\n\n\r");
}
/*
* Task1:
RTX Kernel starts this task with os_sys_init(send_task)
*

task void send_task (void)
{
T_MEAS *mptr;
static unsigned char sInputBuf[MBOX_MSG_BUF_SIZE]; int
cnt=0;
char sSndTskBuf[30]; char
ch;
int MsgFree = 0;
*/
tsk1 =os_tsk_self();
/* get own taskidentificationnumber
#ifndefDISABLE_RECV_TASK
tsk2 = os_tsk_create (rec_task, 0); /* starttask2
#endif /* DISABLE_RECV_TASK*/
os_mbx_init (MsgBox, sizeof(MsgBox));/* initialize the mailbox
os_dly_wait(5);
/* Startup delayforMCB21xx
lcdinit();*/
*/
*/47
clrscr(10);
printstr("
printstr("
MailBox
Simulation
",0,0);
",0,1);
#ifndefDISABLE_RECV_TASK
mptr =_alloc_box(mpool);
/* Allocate a memory for the message
memset(mptr->msgBuf, '\0', MBOX_MSG_BUF_SIZE);
memcpy ( mptr->msgBuf, STD_MSG1, sizeof(STD_MSG1) );
os_mbx_send (MsgBox, mptr, 0xffff); /* Send the message to the mailbox*/
os_dly_wait (100);
mptr = _alloc_box (mpool);

```

```

memset (mptr->msgBuf, '\0', MBOX_MSG_BUF_SIZE);
memcpy ( mptr->msgBuf, STD_MSG2, sizeof(STD_MSG2) );
os_mbx_send (MsgBox, mptr, 0xffff); /* Andsendit.
os_tsk_pass();
/*Cooperativemultitasking
os_dly_wait(100);
*/
*/
*/
mptr = _alloc_box (mpool);
memset (mptr->msgBuf, '\0', MBOX_MSG_BUF_SIZE);
memcpy ( mptr->msgBuf, STD_MSG3, sizeof(STD_MSG3) );
os_mbx_send (MsgBox, mptr, 0xffff); /* Andsendit.
*/
os_dly_wait(100);
#endif /* DISABLE_RECV_TASK */
memset (sInputBuf, '\0', MBOX_MSG_BUF_SIZE);
cnt = 0;
main_menu();w
hile(1)
{
ch = receive(USE_UART);
if(ch == CARRIAGE_RET && cnt > 0)
{
send_string(USE_UART, "\n\n\nr*****SENDING          MAILBOX          USER
MESSAGE*****");
MsgFree = os_mbx_check (MsgBox); if
(MsgFree != 0)
{
mptr = _alloc_box (mpool);
memset (mptr->msgBuf, '\0', MBOX_MSG_BUF_SIZE);
memcpy ( mptr->msgBuf, sInputBuf, strlen((const char *)sInputBuf) );
sprintf (sSndTskBuf, "\nros_mbx_send byTaskID:%d ", tsk1);
*/
send_string(USE_UART, sSndTskBuf); send
_string(USE_UART, "\n\nr");
}
else
{
}
}
#ifdef DISABLE_RECV_TASK
os_mbx_send (MsgBox, mptr, 0xffff); /* And send it. os_dly_wait
(100);
memset (sInputBuf, '\0', MBOX_MSG_BUF_SIZE); cnt
= 0;
send_string(USE_UART, "\n\nrMailbox is FULL");
main_menu();48
#endif /* DISABLE_RECV_TASK */
}

```

```

else if ( KEY_SPACE == ch)
{
MsgFree = os_mbx_check (MsgBox); if
(MsgFree == 0)
{
send_string(USE_UART,"\n\rMailboxisFULL. .... ");
}
else
{
sprintf (sSndTskBuf, "\n\rMailBox Free Count : %d ", MsgFree);
send_string(USE_UART,sSndTskBuf);
send_string(USE_UART,"\n\n\r");
}
os_dly_wait (100);
main_menu();
}
else if ( KEY_ESC == ch)
{
cnt = 0;
memset (sInputBuf,'\0',MBOX_MSG_BUF_SIZE);
send_string(USE_UART,"\n\rClearing Buffer
PleaseWait. .... ");
os_dly_wait (100);
main_menu();
}
else if ('\0' != ch)
{
sInputBuf[cnt++] = ch;
cnt %= MBOX_MSG_BUF_SIZE;
if(ch == CARRIAGE_RET &&cnt==1)
{
cnt = 0;
}
else
{
putcharr (USE_UART,ch);
//emptystring
}
}
}
//
}
os_tsk_delete_self();
/* We are done here, deletethistask
#ifdef DISABLE_RECV_TASK
/*
* Task 2: RTX Kernel starts this task with os_tsk_create (rec_task,0)
*
task void rec_task (void)

```



```

{
T_MEAS *rptr;
static char sRxTskBuf[MBOX_MSG_BUF_SIZE];
for (;;)
{
os_mbx_wait (MsgBox, (void **)&rptr, 0xffff); /* wait for the message
send_string(USE_UART, "\n\n\n
\r*****MAILBOX MESSAGE RECEIVED*****");
*/
*/
*/49
sprintf (sRxTskBuf, "\nrec_task by Task ID: %d ", tsk2); send_string(USE_UART, sRxTskBuf);
memset (sRxTskBuf, '\0', MBOX_MSG_BUF_SIZE);
memcpy      (      sRxTskBuf,      rptr->msgBuf,      strlen(rptr->msgBuf)      );
      send_string(USE_UART, "\n\n\rReceived Mbox: ");
send_string(USE_UART, sRxTskBuf);
send_string(USE_UART, "\nr*****");
_free_box(mpool, rptr);
/* free memory allocated for message
main_menu();
}
}
#endif /* DISABLE_RECV_TASK */
/*
*
Main: Initialize and start RTXKernel
*
int main (void)
{
initserial(USE_UART);
_init_box(mpool, sizeof(mpool),
sizeof(T_MEAS));
os_sys_init(send_task);
*/
*/
/* uart0 initialization*/
/* initialize the 'mpool' memory for
/* the membox dynamic allocation
/* initialize and start task1
*/
*/
*/
}
/*
* end of file
*
*/
/*****/
RTX_CONFIG.C

```

```

/*****
/*
*
RL-ARM -RTX
*
*
*
*
*
*
Name:
RTX_CONFIG.C
Purpose: Configuration of RTX Kernel for NXPLPC21xx
Rev.:
V4.20
This code is part of the RealView Run-TimeLibrary.
Copyright (c) 2004-2011 KEIL - An ARM Company. All rightsreserved.
*
*/
#include <RTL.h>
#include<LPC21xx.H>
/*LPC21xxdefinitions
*/
/*
*
RTX User configuration partBEGIN
*
//----- <<< Use Configuration Wizard in Context Menu >>> -----
//
// <h>TaskConfiguration
//=====
//
*/50
//
<o>Number of concurrent running tasks<0-250>
//
<i> Define max. number of tasks that will run at the sametime.
//
<i> Default: 6
#ifndefOS_TASKCNT
#defineOS_TASKCNT
6
#endif
//
<o>Number of tasks with user-provided stack<0-250>
//
<i> Define the number of tasks that will use a biggerstack.
//

```

```

<i> The memory space for the stack is provided by the user.
//
<i> Default: 0
#ifndef OS_PRIVCNT
#define OS_PRIVCNT
0
#endif
//
<o> Task stack size [bytes] <20-4096:8> <#/4>
//
<i> Set the stack size for tasks which is assigned by the system.
//
<i> Default: 200
#ifndef OS_STKSIZE
#define OS_STKSIZE
50
#endif
// <q> Check for the stack overflow
// =====
// <i> Include the stack checking code for a stack overflow.
// <i> Note that additional code reduces the RTX performance. #ifndef
OS_STKCHECK
#define OS_STKCHECK
1
#endif
// </h>
// <h> Tick Timer Configuration
// =====
//
<o> Hardware timer <0=> Timer 0 <1=> Timer 1
//
<i> Define the on-chip timer used as a time-base for RTX.
//
<i> Default: Timer 0
#ifndef OS_TIMER
#define OS_TIMER
1
#endif
//
<o> Timer clock value [Hz] <1-1000000000>
//
<i> Set the timer clock value for selected timer.
//
<i> Default: 15000000
(15MHz at 60MHz CCLK and VPBDIV = 4)
#ifndef OS_CLOCK
#define OS_CLOCK
15000000
#endif

```

```

//
<o>Timer tick value [us]<1-1000000>
//
<i> Set the timer tick value for selectedtimer.
//
<i>Default:10000
(10ms)
#ifndefOS_TICK
#defineOS_TICK
10000
#endif
// </h>
// <h>SystemConfiguration
//=====
// <e>Round-Robin Taskswitching
//=====
// <i> Enable Round-Robin Task switching. #ifndef
OS_ROBIN
#defineOS_ROBIN
151
#endif
//
<o>Round-Robin Timeout [ticks]<1-1000>
//
<i> Define how long a task will execute before a taskswitch.
//
<i> Default: 5
#ifndefOS_ROBINTOUT
#defineOS_ROBINTOUT
5
#endif
// </e>
//
<o>Number of user timers<0-250>
//
<i> Define max. number of user timers that will run at the sametime.
//
<i>Default:0
(User timersdisabled)
#ifndefOS_TIMERCNT
#defineOS_TIMERCNT
0
#endif
//
<o>ISR FIFOQueuesize<4=>
4entries
//
<12=>12entries
//

```

```

<24=>24entries
//
<48=>48entries
//
<96=> 96entries
//
<i> ISR functions store requests to thisbuffer,
//
<i> when they are called from the IRQhandler.
//
<i> Default: 16 entries
#ifndefOS_FIFOSZ
#defineOS_FIFOSZ
16
#endif
<8=>
8entries
<16=> 16entries
<32=> 32entries
<64=> 64entries
// </h>
//----- <<< end of configuration section >>> -----
// Standard library systemmutexes
//=====
//
Define max. number system mutexes that are used toprotect
//
the arm standard runtime library. For microlib they are notused.
#ifndefOS_MUTEXCNT
#defineOS_MUTEXCNT
8
#endif
/*
*
RTX User configuration partEND
*
#if
(OS_TIMER==0)
#defineOS_TID_
#defineTIMx(reg)
#elif (OS_TIMER==1)
#defineOS_TID_
#defineTIMx(reg)
#else
#error OS_TIMER invalid
#endif
#defineOS_TIM_
#defineOS_TRV
#defineOS_TVAL

```

```

#define OS_TOVF
#define OS_TFIRQ()
#define OS_TIACK()
*/
4
T0##reg
5
T1##reg
/*Timer0
/* TimerID*/
*/
/*Timer1
/* TimerID*/
*/
(1<<OS_TID_)
/* InterruptMask
((U32)(((double)OS_CLOCK*(double)OS_TICK)/1E6)-1)
TIMx(TC)
/* TimerValue
(TIMx(IR)&1)
/*
OverflowFlag
VICSoftInt
=OS_TIM_;
/* Force Interrupt */
TIMx(IR)=1;
/* InterruptAck
*/
*/
*/
*/ \52
#define OS_TINIT()
VICSoftIntClr = OS_TIM_;
VICVectAddr
= 0;
/* Initialization
TIMx(MR0) =OS_TRV;
TIMx(MCR) =3;
TIMx(TCR) =1;
VICDefVectAddr = (U32)os_def_interrupt;
VICVectAddr15 =(U32)os_clock_interrupt;
VICVectCntl15
= 0x20 |OS_TID_;
\
*/ \
\
\
\
\

```

```

#define OS_IACK()VICVectAddr=0; /* InterruptAck*/
#define OS_LOCK()
define
OS_UNLOCK
()VICIntEnClr
VICIntEnable=OS_TIM_;
=OS_TIM_; /* Task Lock
/* Task Unlock*/
*/
/* WARNING: Using IDLE mode might cause you troubles while debugging.*/ #define_idle_()
PCON =1;
/*
*
GlobalFunctions
*
*/
/*
os_idle_demon
task void os_idle_demon (void) {
/* The idle demon is a system task, running when no other task is ready*/
/* to run. The 'os_xxx' function calls are not allowed fromthistask.
*/
*/
for (;;) {
/* HERE: include optional user code to be executed when no task runs.*/
}
}
/*
os_tmr_call
void os_tmr_call (U16 info) {
/* This function is called when the user timer hasexpired.Parameter
/* 'info' holds the value, defined when the timerwascreated.
*/
*/
*/
/* HERE: include optional user code to be executed on timeout. */
}
/*
os_error
void os_error (U32 err_code){
/* This function is called when a runtime error is detected. Parameter*/
/* 'err_code' holds the runtime error code (definedinRTL.H).
*/
*/
/* HERE: include optional code to be executed on runtime error. */ for (;;)
}
/*
*
RTX ConfigurationFunctions

```

```

*
static void os_def_interrupt(void)irq
{
/* Default Interrupt Function: may be called when timer ISR is disabled */ OS_IACK();
}
*/53
#include <RTX_lib.c>
/*
* end of file
*
*
/*****
LCD.C
*****/
#include<LPC214x.H>
#defineRS
0x00000400
#defineCE
0x00001000
/* LPC214x definitions */
/* P0.10*/
/* P1.11*/
#define SET1
#define OFF0
void lcdcmd(char cmd); void
lcdat(char cmd);
void printstr(unsigned char *str, char x, char y);
voidwait(void)
{
d;
for (d = 0; d <100000;d++);
}
void lcdinit(void)
{
IODIR0 |= 0x000014f0;
lcdcmd(0x28);
lcdcmd(0x28);
lcdcmd(0x0c);
lcdcmd(0x06);
lcdcmd(0x01);
lcdcmd(0x0f);
wait();/(1600);
}
void gotoxy(char x, char y)
{
if(y == 0)
lcdcmd(0x80+x);
else
lcdcmd(0xc0+x);
}

```



```

}
void printstr(unsigned char *str, char x, char y)
{
char i; gotoxy(x,y);
wait();//(500);
for(i=0;str[i]!='\0';i++)lcddat(str[i
]);
}
/* wait function */ int
/* only to delay for LED flashes*/54
void lcdcmd(charcmd)
{
unsigned charLCDDAT;
LCDDAT = (cmd&0xf0);
IOSET0 |=LCDDAT;
IOCLR0 |= RS;
IOSET0 |= CE;
wait();//(100);
IOCLR0 |= CE;
IOCLR0 |= 0X0000FFF;
//higher nibble
//enablelcd
LCDDAT = ((cmd<<0x04)&0xf0);
IOSET0 |=LCDDAT;
IOCLR0 |= RS;
IOSET0 |= CE;
wait();//(100);
IOCLR0 |= CE;
IOCLR0 |= 0X0000FFF;
//lower nibble
//enablelcd
}
void lcddat(char cmd)
{
unsigned char LCDDAT;
LCDDAT = (cmd&0xf0);
IOSET0 |=LCDDAT;
IOSET0 |= RS;
IOSET0 |= CE;
wait();//(100);
IOCLR0 |= CE;
IOCLR0 |= 0X0000FFF;
//higher nibble
//enablelcd
LCDDAT = ((cmd<<0x04)&0xf0);
IOSET0 |=LCDDAT;
IOSET0 |= RS;
IOSET0 |= CE;
wait();//(100);

```

```

IOCLR0 |= CE;
IOCLR0 |= 0X00000FFF;
//lower nibble
//enablelcd
}
void clrscr(char ch)
{
if(ch==0)
{
printstr("
gotoxy(0,0);
}
else if(ch ==1)
{
printstr("
gotoxy(0,1);
}
else
{
lcdcmd(0x01);
//delay(100);
}
}
",0,0);
",0,1);55
/*****

```

UART.C

```

/*****
/* This file contains driver functions to send and receive data via uart0inthe
/* ARM LPC2148 Development board itself */
/*****/
#include
<LPC214x.H>
#include "config.h"
#define TEMT (1<<6)
void initserial(unsigned char uart)
{
if(0 == uart)
{
PINSEL0=0x00000005;
forUART0*/
U0LCR=0x83;
U0FDR=0x00000010;
U0DLL=98;
*/
U0LCR=0x03;
U0IER=0x01;
/* Make pins 19 and 21 to function as TXD0and RXD0
/* 8 bits, no Parity, 1Stopbit

```

```

/* DIVADDVAL = 0; MULVAL = 1*/
/* 9600 Baud Rate @ 15MHz VPB Clock; =97.65=98
*/
/* DLAB = 0*/
/* Enable reciever data availableinterrupt*/
}
else
{
PINSEL0=0x00050000;
RXD0 for UART0*/
U1LCR=0x83;
U1FDR=0x00000010;
U1DLL=98;
*/
U1LCR=0x03;
U1IER=0x01;
/* Make pins 19 and 21 to function as TXD0and
/* 8 bits, no Parity, 1Stopbit
*/
/* DIVADDVAL = 0; MULVAL = 1*/
/* 9600 Baud Rate @ 15MHz VPB Clock; =97.65=98
/* DLAB = 0*/
/* Enable reciever data availableinterrupt*/
}
}
void putcharr (unsigned char uart, unsignedcharch)
SerialPort*/
{
if(0 == uart)
{
while (!(U0LSR &TEMT)); U0THR
=ch;
}
else
{
while (!(U1LSR &TEMT)); U1THR
=ch;
/* Writes character to
}
}
unsigned char getcharr (unsignedcharuart)
SerialPort*/
{
if(0 == uart)
{
while (!(U0LSR & 0x01));
return (U0RBR);
/* Reads character from
*/56

```

```

}
else
{
while (!(U1LSR & 0x01));
return (U1RBR);
}
}
char receive(unsigned char uart) /*function for receiving data from sensor (readsbyte by byte &
returns value if
exist,else#)
*/
{
if(0 == uart)
{
if (U0LSR&0x01)
/* If U0LSR 1st bit contains valid data, thenreturn value ofU0RBR*/
{
return (U0RBR);
}
return'\0';
/* If other than 0 to 9 data is recievedreturn
##*/
}
else
{
if (U1LSR&0x01)
/* If U0LSR 1st bit contains valid data, thenreturn
value of U0RBR*/
{
return (U1RBR);
}
return'\0';
/* If other than 0 to 9 data is recievedreturn
##*/
}
}
void send_string(unsigned charuart,char*cpr)
{
while(*cpr != '\0')
{
putcharr (uart,*cpr); cpr++;
}
}
unsigned char* receive_string(unsignedcharuart)
port*/
{
static unsigned char c[30]; unsigned
char i=0;
c[i] = getcharr(uart); while(c[i] !=

```

```

CARRIAGE_RET)
{
i++;
c[i] = getcharr(uart);
}
c[i] = '\0';
return(c);
}
/* Writes string to serial port*/
/* Reads string to serial57
CONFIG.H
/*****
#define
MAILBOX_MEMORY_POOL_CNT16
#define
MBOX_MSG_BUF_SIZE
100
#defineUSE_UART
0
/*****
/*
Enable below macro to disable the the recv task to check mailbox full*/
//#define
DISABLE_RECV_TASK
/*****
#define STD_MSG1 "MailBox Test Message 1" #define
STD_MSG2 "MailBox Test Message 2" #define STD_MSG3
"MailBox Test Message3"
#define LINE_FEED 0x0A #define
CARRIAGE_RET 0x0D #define
KEY_SPACE
0x20
#define
KEY_ESC0x1B
MAIL PROGRAM PROGRAM
PORTDETAILS UART0

```

ARM

DETAILS

P0.0	TXDO
P0.1	RXDO

UART1

ARM

DETAILS

P0.8	TXD1
P0.9	RXD1

LCD PORT DETAILS

ARM

PO.10

P1.11

DETAILS

RS LCD PIN

CE LCD PIN

Post Lab Questions:

1. Mention the operations that can be performed on a mailbox.
2. When does the mailbox will get deleted?
3. Illustrate the operation of reading operation from a mailbox.
4. How to configure the mailbox?
5. What is branch prediction?

Result:

The C-Language program to create a mailbox and to understand the about the RTOS functions is developed and is verified.

Exp. No.: 11

BLINKING OF AN LED USING ARDUINO

Date:

Aim:

To interface an LED with Arduino and to write a program to make the LED blink at defined intervals.

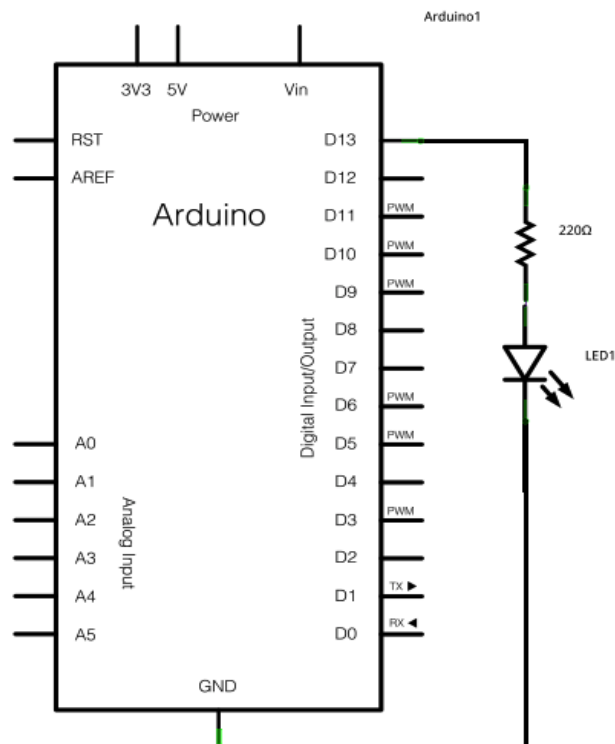
Components required:

- PC or laptop
- Arduino UNO
- 220 Ω resistor
- LED
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Connections:



- Connect the longer leg of the LED, that is the anode, to the digital pin 13 in the Arduino, through a 220 Ω resistor.
- Connect the shorter leg of the LED, that is the cathode, to the ground (GND) pin on the Arduino.

Procedure:

1. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
2. Open Aurdino IDE and open a new editor.
3. Enter the required code save it as a “ino” file.
4. In the “Tools” menu go to the “Port” tab and select the appropriate port.
5. Upload the program into the Arduino and verify the output.

Program:

To make the built-in LED blink:

```
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000);                    // wait for a second  
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW  
  delay(1000);                    // wait for a second  
}
```

To make the externally connected LED blink:

```
int led = 13;  
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(led, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000);             // wait for a second  
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW  
  delay(1000);             // wait for a second  
}
```

Result:

A program has been written to make the built-in and externally connected LED blink and the output verified.

Exp. No.: 12

FADING OF AN LED USING ARDUINO

Date:

Aim:

To interface an LED with Arduino and to write a program to make the LED fade.

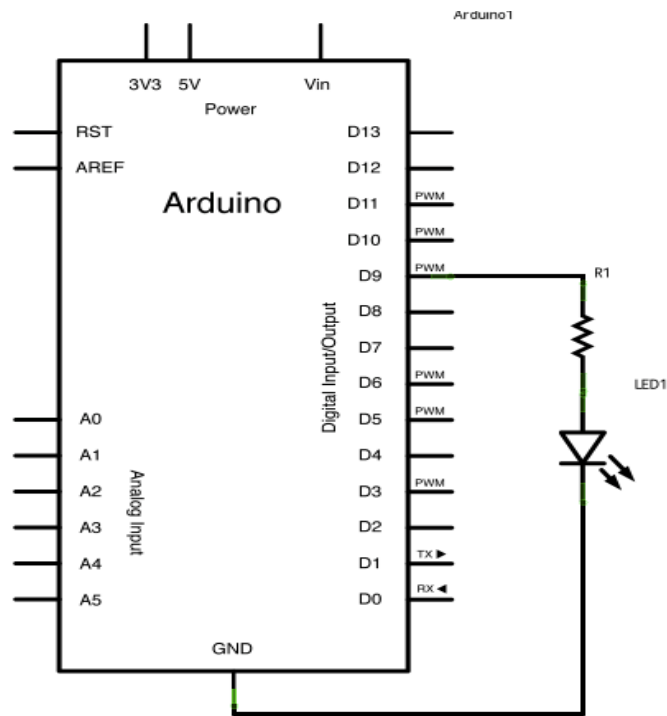
Components required:

- PC or laptop
- Arduino UNO
- 220 Ω resistor
- LED
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Connections:



- Connect the longer leg of the LED, that is the anode, to the digital pin 9 in the Arduino, through a 220 Ω resistor.
- Connect the shorter leg of the LED, that is the cathode, to the ground (GND) pin on the Arduino.

Procedure:

1. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
2. Open Arduino IDE and open a new editor.
3. Enter the required code save it as a “ino” file.
4. In the “Tools” menu go to the “Port” tab and select the appropriate port.

5. Upload the program into the Arduino and verify the output.

Program:

```
int led = 9;          // the PWM pin the LED is attached to
int brightness = 0;   // how bright the LED is
int fadeAmount = 5;   // how many points to fade the LED by
// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);
  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;
  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Result:

A program has been written to make the LED connected to an Arduino fade and the output verified.

Exp. No.: 13

TURNING OF AN LED ON AND OFF USING A PUSH BUTTON

Date:

To interface a pushbutton with an Arduino and to write a program to turn the built-in LED on and off using the pushbutton.

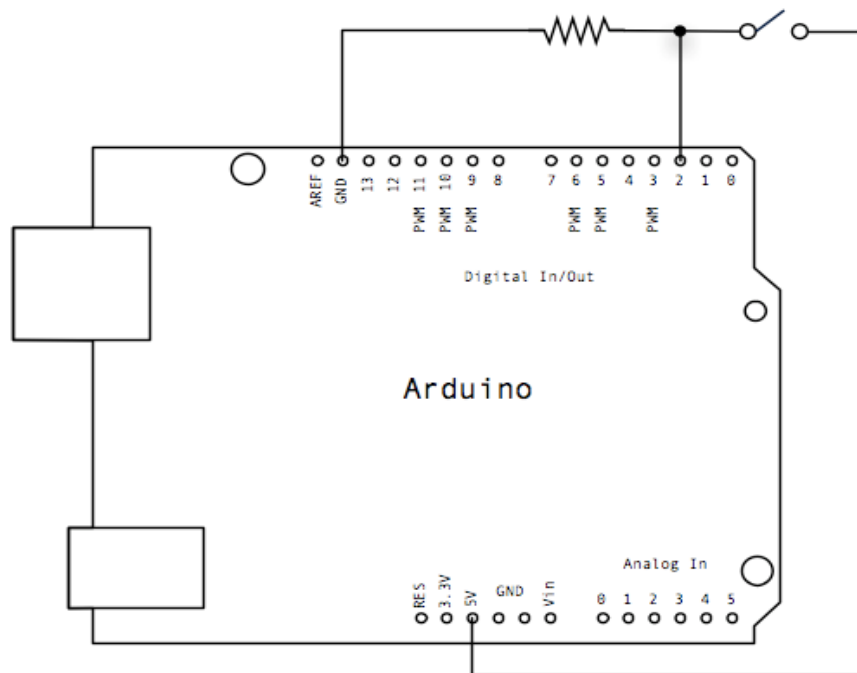
Components required:

- PC or laptop
- Arduino UNO
- 10 k Ω resistor
- Pushbutton
- Breadboard and connecting wires

Software required:

- Arduino IDE

Connections:



- Connect digital pin 2 on the Arduino to one leg of the pushbutton.
- Connect the same leg through a 10 k Ω resistor to the ground (GND) pin on the Arduino.
- Connect the other leg of the pushbutton to the 5V supply on the Arduino.

Procedure:

1. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
2. Open Arduino IDE and open a new editor.

3. Enter the required code save it as an “ino” file.
4. In the “Tools” menu go to the “Port” tab and select the appropriate port.
5. Upload the program into the Arduino and verify the output.

Program:

```
// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;  // the number of the pushbutton pin
const int ledPin = 13;    // the number of the LED pin
// variables will change:
int buttonState = 0;       // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Result:

A pushbutton has been interfaced with an Arduino and a program written, to turn the built-in LED on and off, has been verified.

Exp. No.: 14 INTERFACING A WATER-LEVEL SENSOR WITH AN ARDUINO

Date:

Aim:

To interface a water-level sensor with an Arduino and to write a program to measure the water level.

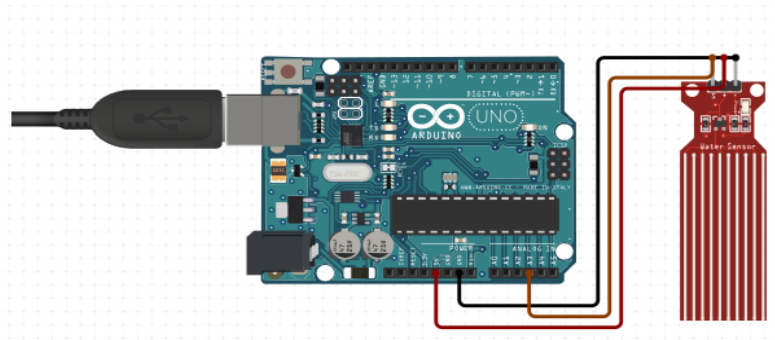
Components required:

- PC or laptop
- Arduino UNO
- Water-level sensor
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Connections:



- Connect the GND pin on the water-level sensor to that on the Arduino.
- Connect the SIG pin on the sensor to analog pin A3 of the Arduino.
- Connect the VCC pin on the sensor to the 5V pin on the Arduino.

Procedure:

1. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
2. Open Arduino IDE and open a new editor.
3. Enter the required code save it as an “ino” file.
4. In the “Tools” menu go to the “Port” tab and select the appropriate port.
5. Upload the program into the Arduino and verify the output on the serial monitor in Arduino IDE.

Program:

```
int sensorPin = A3;  
int sensorValue = 0;  
int value;  
void setup() {  
  // put your setup code here, to run once:  
  Serial.begin(9600);
```

```

pinMode(sensorPin, INPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  //sensorValue = analogRead(sensorPin);
  value = analogRead(sensorPin);
  if (value<=480){
    Serial.println("Water level: 0mm - Empty!");
  }
  else if (value>480 && value<=530){
    Serial.println("Water level: 0mm to 5mm");
  }
  else if (value>530 && value<=615){
    Serial.println("Water level: 5mm to 10mm");
  }
  else if (value>615 && value<=660){
    Serial.println("Water level: 10mm to 15mm");
  }
  else if (value>660 && value<=680){
    Serial.println("Water level: 15mm to 20mm");
  }
  else if (value>680 && value<=690){
    Serial.println("Water level: 20mm to 25mm");
  }
  else if (value>690 && value<=700){
    Serial.println("Water level: 25mm to 30mm");
  }
  else if (value>700 && value<=705){
    Serial.println("Water level: 30mm to 35mm");
  }
  else if (value>705){
    Serial.println("Water level: 35mm to 40mm");
  }
  delay(2000);
}

```

Output:

Result:

A water-level sensor was interfaced with an Arduino and a program to measure the water level was written and executed.

Exp. No.: 15

INTERFACING AN ULTRASONIC SENSOR WITH AN ARDUINO

Date:

Aim:

To interface an ultrasonic sensor with an Arduino and to write a program to measure the distance of an object from the sensor.

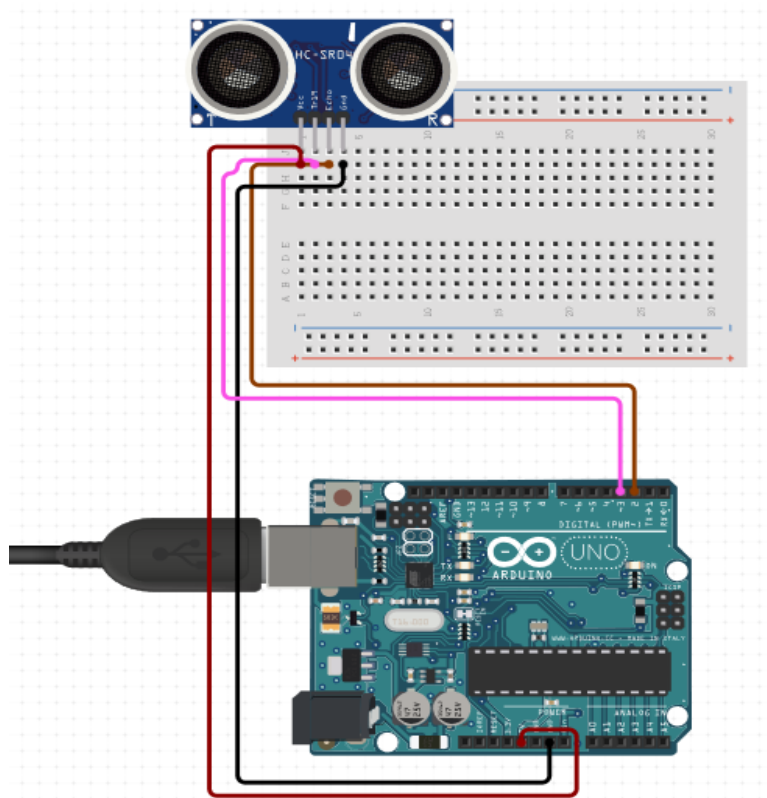
Components required:

- PC or laptop
- Arduino UNO
- Ultrasonic sensor HCSR04
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Connections:



- Connect the ECHO pin on the HCSR04 sensor to digital pin 2 of the Arduino.
- Connect the ground (GND) of the sensor to GND of the Arduino.
- Connect the TRIG pin of the sensor to digital pin 3 of the Arduino.
- Connect the VCC pin of the sensor to the 5V pin on the Arduino.

Procedure:

1. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
2. Open Arduino IDE and open a new editor.

3. Enter the required code save it as an “ino” file.
4. In the “Tools” menu go to the “Port” tab and select the appropriate port.
5. Upload the program into the Arduino and verify the output on the serial monitor in Arduino IDE.

Program:

```
// Include Libraries
#include "Arduino.h"
#include "NewPing.h"

// Pin Definitions
#define HCSR04_PIN_TRIG 3
#define HCSR04_PIN_ECHO 2

// Global variables and defines
// object initialization
NewPing hcsr04(HCSR04_PIN_TRIG,HCSR04_PIN_ECHO);

// define vars for testing menu
const int timeout = 10000;    //define timeout of 10 sec
char menuOption = 0;
long time0;
// Setup the essentials for your circuit to work. It runs first every time your circuit is powered with
// electricity.
void setup()
{

    // Setup Serial which is useful for debugging
    // Use the Serial Monitor to view printed messages
    Serial.begin(9600);
    while (!Serial) ; // wait for serial port to connect. Needed for native USB
    Serial.println("start");
    menuOption = menu();
}

// Main logic of your circuit. It defines the interaction between the components you selected. After
// setup, it runs over and over again, in an eternal loop.
void loop()
{
    if(menuOption == '1') {
        // Ultrasonic Sensor - HC-SR04 - Test Code
        // Read distance measurment from UltraSonic sensor
        int hcsr04Dist = hcsr04.ping_cm();
        delay(1001);
        Serial.print(F("Distance: ")); Serial.print(hcsr04Dist); Serial.println(F("[cm]"));
    }
    if (millis() - time0 > timeout)
    {
        menuOption = menu();
    }
}
```



```

    }
}

// Menu function for selecting the components to be tested
// Follow serial monitor for instructions
char menu()
{
    Serial.println(F("\nWhich component would you like to test?"));
    Serial.println(F("(1) Ultrasonic Sensor - HC-SR04"));
    Serial.println(F("(menu) send anything else or press on board reset button\n"));

    while (!Serial.available());

    // Read data from serial monitor if received
    while (Serial.available())
    {
        char c = Serial.read();
        if (isAlphaNumeric(c))
        {
            if(c == '1')
                Serial.println(F("Now Testing Ultrasonic Sensor - HC-SR04"));
            else
            {
                Serial.println(F("illegal input!"));
                return 0;
            }
            time0 = millis();
            return c;
        }
    }
}

```

Output:

Result:

An ultrasonic sensor has been interfaced with the Arduino and the distance of an object from the sensor has been measured.

Exp. No.: 16

TO USE A BUZZER (OR PIEZO SPEAKER) WITH ARDUINO

Date:

Aim:

To interface an buzzer sensor with an Arduino.

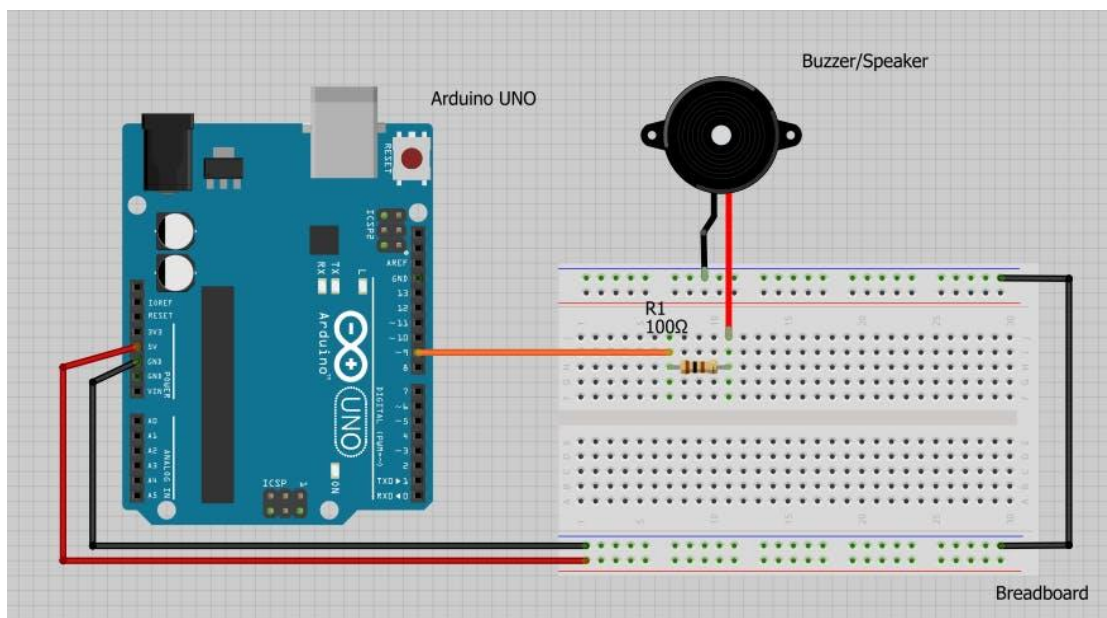
Components required:

- PC or laptop
- Arduino UNO
- Buzzer
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Connections:



Procedure:

6. Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
7. Open Arduino IDE and open a new editor.
8. Enter the required code save it as an “ino” file.
9. In the “Tools” menu go to the “Port” tab and select the appropriate port.
10. Upload the program into the Arduino and verify the output on the serial monitor in Arduino IDE.

Program:

```
const int buzzer = 9; //buzzer to arduino pin 9
void setup(){
  pinMode(buzzer, OUTPUT); // Set buzzer - pin 9 as an output
}
void loop(){
  tone(buzzer, 1000); // Send 1KHz sound signal...
  delay(1000);      // ...for 1 sec
  noTone(buzzer);   // Stop sound...
  delay(1000);      // ...for 1sec
}
```

Result:

Thus the Buzzer Sensor (Piezo Speaker) is successfully interfaced and the performance is studied.

Exp. No.: 17

MQ-6 GAS SENSOR INTERFACING WITH ARDUINO

Date:

Aim:

To interface a MQ-6 Gas sensor with an Arduino.

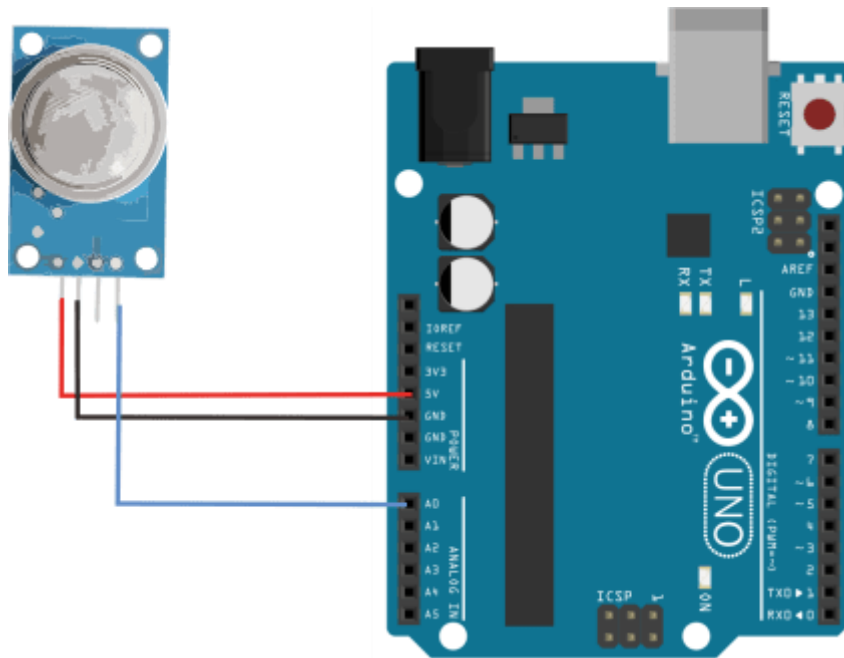
Components required:

- PC or laptop
- Arduino UNO
- MQ-6 GAS SENSOR
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Connections:



VCC - 5V
GND - GND
S - Analog pin0

Procedure:

- Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
- Open Arduino IDE and open a new editor.
- Enter the required code save it as an “ino” file.
- In the “Tools” menu go to the “Port” tab and select the appropriate port.
- Upload the program into the Arduino and verify the output on the serial monitor in Arduino IDE.

Program:

```
void setup() {  
  // initialize serial communication at 9600 bits per second:  
  Serial.begin(9600);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
  // read the input on analog pin 0:  
  int sensorValue = analogRead(A0);  
  // print out the value you read:  
  Serial.println(sensorValue);  
  delay(1000);  
}
```

Result:

Thus the MQ-6 Gas Sensor is interfaced with Arduino and the performance is studied.

ExpNo:18

Implementing zigbee protocol with ARM

Date:

Aim

To write C Programs for Zigbee Protocol and verify the communication between Xbee Module Transmitter and Receiver.

Pre Lab Questions:

1. What are the applications of zigbee protocol?
2. Why Zigbee based is preferred for wireless communication?
3. What is the function of a scheduler?
4. What is the main function of voltage convertors in UART?
5. List the advantages of using Zigbee protocol.

Apparatus & Software Required:

1. LPC2148 Development board.
2. Keil μ Vision5 software.
3. Flash Magic.
4. USB cable.
5. Zigbee Module Tx and Rx.

Theory:

The X Bee/X Bee-PRO ZNet 2.5 (formerly known as Series 2 and Series 2 PRO) RF Modules were directed to operate within the ZigBee protocol. The modules provide reliable delivery of data between remote devices. Zigbee is the communication protocol like wifi and Bluetooth. Xbee is the module using Zigbee protocol

Some of its features are:

ZigBee is targeted at radio-frequency (RF) applications

Low data rate, long battery life, and secure networking

Transmission range is between 10 and 75 meters (33~246 feet)

The addressing space allows of extreme node density—

up to 18,450,000,000,000,000,000 devices (64 bit IEEE address)

Using local addressing, simple networks of more than 65,000 nodes can be configured, with reduced address overhead

The radios use direct-sequence spread spectrum coding, which is managed by the digital stream into the modulator.

To ensure reliable data transmission

Binary phase shift keying (BPSK) in the 868/915 MHz

Offset quadrature phase shift keying (O-QPSK) at 2.4 GHz

Procedure

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash

Magic Software.

*****/

ARM TRANSMITTER

PROGRAM LCD.C

*****/

```

#include <LPC214x.h>
#include "lcd.h"
#define RS
0x00000400
#define CE
0x00001800
/* P0.10 */
/* P1.11 */
/*void clrscr(char ch); void
lcdinit(void);
voidlcdcmd(char);
voidlcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or1
voidprintstr(char*,char,char);
//string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number);*/
#define SET1
#define OFF0
unsigned int thousands,hundreds,tens,ones; void wait
(void)
{
int
d;
for (d = 0; d <100000;d++);
}
/* wait function*/
void lcdinit()
{
IODIR0 |= 0xFFFFFFFF;
IOCLR0 |= 0X00000FFF;
lcdcmd(0x28);lcd
cmd(0x28);
lcdcmd(0x0c);
lcdcmd(0x06);
lcdcmd(0x01);
lcdcmd(0x0f);
wait();
}
void gotoxy(char x, char y)
{
if(y == 0)
lcdcmd(0x80+x);
else
lcdcmd(0xc0+x);
}
void printstr(char *str, char x, char y)
{
char i; gotoxy(x,y);

```

```

wait();//(500);
for(i=0;str[i]!='\0';i++)
/* only to delay for LED flashes*/81
lcddat(str[i]);
}
void lcdcmd(charcmd)
{
unsigned charLCDDAT;
LCDDAT = (cmd&0xf0);
IOSET0 =LCDDAT;
IOCLR0 = RS;
IOSET0 = CE;
wait();
//higher nibble
//(100);
//enable lcd
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
LCDDAT = ((cmd<<0x04)&0xf0);
IOSET0 =LCDDAT;
IOCLR0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
}
void lcddat(char cmd)
{
unsigned charLCDDAT;
LCDDAT = (cmd&0xf0);
IOSET0 =LCDDAT;
IOSET0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
//lower nibble
//enablelcd
//higher nibble
//enablelcd
LCDDAT = ((cmd<<0x04)&0xf0);
IOSET0 =LCDDAT;
IOSET0 = RS;
IOSET0 = CE;
wait();//(100);
IOCLR0 = CE;
IOCLR0 = 0X00000FFF;
//lower nibble
//enablelcd

```



```

}
void clrscr(char ch)
{
if(ch==0)
{
printstr("
gotoxy(0,0);
}
else if(ch ==1)
{
printstr("
gotoxy(0,1);
}
else
{
lcdcmd(0x01);
//delay(100);
}
",0,0);
",0,1);82
}
void split_numbers(unsigned int number)
{
thousands = (number /1000);
number %= 1000;
hundreds = (number / 100);
number %= 100;
tens = (number / 10);
number %= 10;
ones = number ;
}
void Wait_Msg(void)
{
lcdcmd(0x01);
printstr("
Please Wait ", 0,0);
}
void Welcome_Msg(void)
{
lcdcmd(0x01);
printstr("
Welcometo
", 0,0);
printstr("
SMMICRRO
", 0,1);
}
/*****/
LCD.h

```

```

/*****/
void clrscr(char ch); void
lcdinit(void); void
lcdcmd(char); void
lcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or 1
void printstr(char*,char,char);
//string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number); void
Wait_Msg(void);
void Welcome_Msg(void);
/*****/
UART_1.C
/*****/
#include <LPC214X.H>
#include "lcd.c"
#define TEMT0X40
void uart_1(void);
void delay(void);
void putcharr (unsigned char ch);
void tx_string(char str);
int main(void)
{
    uart_1();
    lcdinit();
    delay();
    /* Writes character to Serial Port */
    delay();
    delay();
    printstr("SM MICRO SYSTEM",0,0);
    while(1)
    {
        tx_string('C');
        gotoxy(7,1);
        lcddat('C');
        delay();
        delay();
        delay();
        delay();
        while(1);
    }
}
void uart_1(void)
{
    PINSEL0 = 0x00050000;
    U1LCR = 0x83;
    U1FDR = 0x00000010;
}

```

```

U1DLL = 98;
U1LCR = 0x03;
U1IER = 0x01;
}
void delay(void)
{
int
d;
for (d = 0; d <100000;d++);
}
/* only to delay for LED flashes*/
void tx_string(char str)
{
putcharr(str);
}
void putcharr (unsignedcharch)
{
while (!(U1LSR&TEMT));
*/
U1THR = ch;
}
/* Writes character to SerialPort*/
/* U1LSR --> Statusregister
/*****
ARM RECEIVERPROGRAM
/*****
#include <LPC214X.H>
#include "lcd.c"
void uart_1(void); void
delay(void);
unsigned chargetcharr(void);
/* Reads character from SerialPort*/
int main(void)
{
char rx_data;
uart_1();
lcdinit();
printstr("SM MICRRO SYSTEM",0,0);
while(1)
{
84
rx_data=getcharr();
Port*/
}
/* Reads character fromSerial
gotoxy(7,1);
lcddat(rx_data);
}
voiduart_1(void) /* UART Installation*/
{

```

```

PINSEL0 = 0x00050000;
U1LCR = 0x83;
U1FDR = 0x00000010;
U1DLL = 98;
U1LCR = 0x03;
U1IER = 0x01;
}
void delay(void)
{
int
d;
for (d = 0; d <100000;d++);
}
unsigned char getcharr(void)
{
while (!(U1LSR & 0x01));
return (U1RBR);
/* only to delay for LED flashes*/
/* Reads character from SerialPort*/
}

```

Implementing zigbee protocol with ARM PROGRAMS PORT DETAIL

TRANSMITTER PROGRAM

ARM	Details
P0.8	TXD1
P0.9	RXD1
P0.10	RS LCD PIN
P1.11	CE LCD PIN

RECEIVER PROGRAM

ARM	Details
P0.8	TXD1
P0.9	RXD1
P0.10	RS LCD PIN
P1.11	CE LCD PIN

Post Lab Questions:

- 1.How to verify the communication between Transmitter and Receiver?
- 2.Which module is using Zigbee protocol?
- 3.How many UART ports available in LPC2148?
- 4.Write the two modes of communication are used in a ZigBee network.
- 5.Mention the transmission range for Zigbee protocol.

Result:

The C-Language program for Zigbee Protocol is written and the communication between Xbee Module Transmitter and Receiver is verified.

Exp. No.: 19.A

LCD INTERFACING WITH THE ARDUINO MODULE

Date:

Aim:

To interface a LCD with an Arduino.

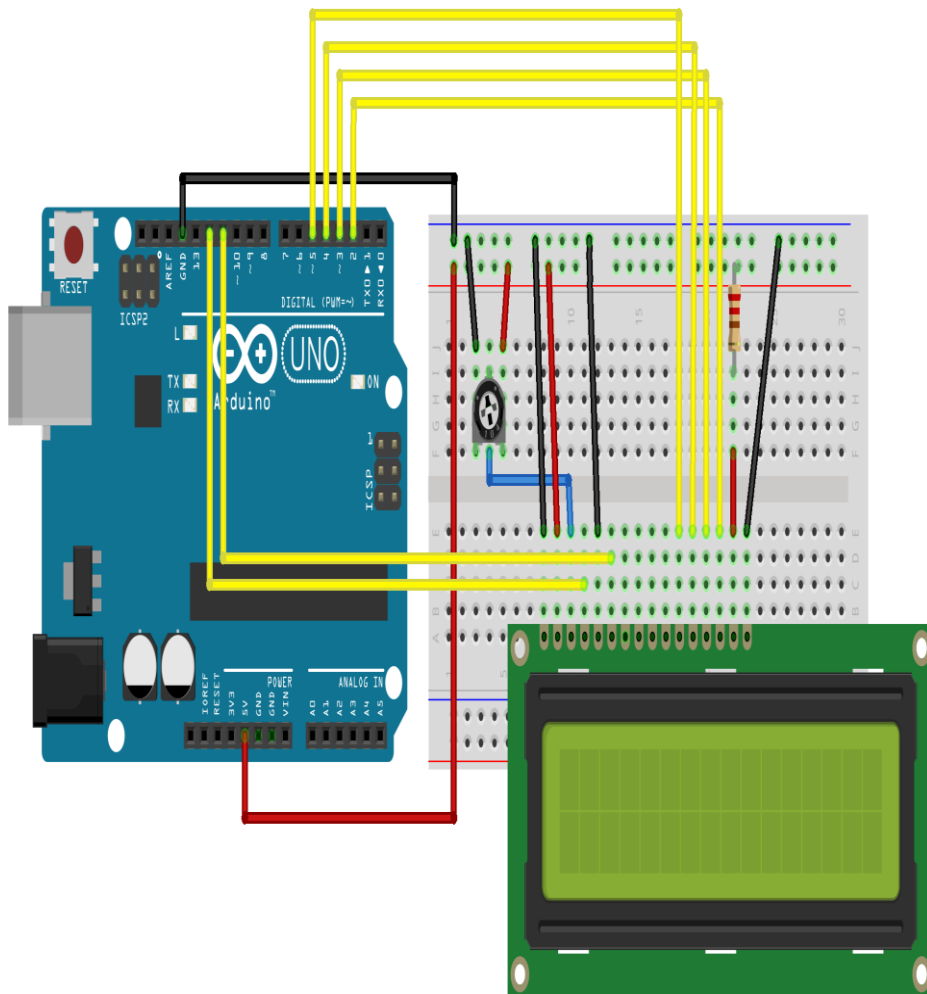
Components required:

- PC or laptop
- Arduino UNO
- LCD module
- Breadboard
- Connecting wires

Software required:

- Arduino IDE

Connections:



- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4

- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2
- Additionally, wire a 10k pot to +5V and GND, with it's wiper (output) to LCD screens VO pin (pin3). A 220 ohm resistor is used to power the backlight of the display, usually on pin 15 and 16 of the LCD connector

Procedure:

- Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
- Open Arduino IDE and open a new editor.
- Enter the required code save it as an “ino” file.
- In the “Tools” menu go to the “Port” tab and select the appropriate port.
- Upload the program into the Arduino and verify the output on the serial monitor in Arduino IDE.

Program:

```
#include <LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

Result:

Thus the LCD module is interfaced with Arduino and studied.

Exp. No.: 19.B

RFID MODULE INTERFACING WITH ARDUINO

Date:

Aim:

To interface a RFID module with an Arduino.

Components required:

- PC or laptop
- Arduino UNO
- RFID module
- Breadboard
- Connecting wires

RFID means radio-frequency identification. RFID uses electromagnetic fields to transfer data over short distances. RFID is useful to identify people, to make transactions, etc...

You can use an RFID system to open a door. For example, only the person with the right information on his card is allowed to enter. An RFID system uses:

- tags attached to the object to be identified, in this example we have a keychain and an electromagnetic card. Each tag has his own identification (UID).



- two-way radio transmitter-receiver, the reader, that send a signal to the tag and read its response.



Specifications:

- Input voltage: 3.3V
- Price: approximately 3\$ (check best price on Maker Advisor)
- Frequency: 13.56MHz

Library download

Here's the library you need for this project:

1. Download the RFID library here created by miguelbalboa
2. Unzip the RFID library
3. Install the RFID library in your Arduino IDE
4. Restart your Arduino IDE

Pin wiring

Pin	Wiring to Arduino Uno
-----	-----------------------

SDA	Digital 10
SCK	Digital 13
MOSI	Digital 11
MISO	Digital 12
IRQ	unconnected
GND	GND
RST	Digital 9
3.3V	3.3V

Caution: You must power this device to 3.3V!

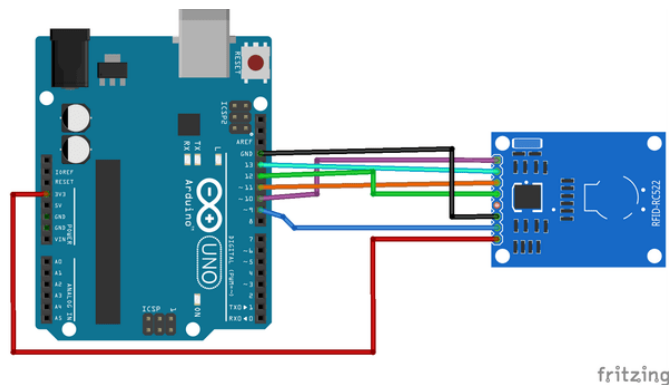
Software required:

- Arduino IDE

Procedure:

- Connect the components on a breadboard as mentioned above and connect the Arduino to the PC/laptop.
- Open Arduino IDE and open a new editor.
- Enter the required code save it as an “ino” file.
- In the “Tools” menu go to the “Port” tab and select the appropriate port.
- Upload the program into the Arduino and verify the output on the serial monitor in Arduino IDE.

Connections:



Program:

```
#include <SPI.h>
#include <MFRC522.h>
```

```

#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.

void setup()
{
  Serial.begin(9600); // Initiate a serial communication
  SPI.begin(); // Initiate SPI bus
  mfrc522.PCD_Init(); // Initiate MFRC522
  Serial.println("Approximate your card to the reader...");
  Serial.println();
}

void loop()
{
  // Look for new cards
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
  //Show UID on serial monitor
  Serial.print("UID tag :");
  String content= "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("Message : ");
  content.toUpperCase();
  if (content.substring(1) == "BD 31 15 2B") //change here the UID of the card/cards that you want
  to give access
  {
    Serial.println("Authorized access");
    Serial.println();
    delay(3000);
  }
  else {
    Serial.println(" Access denied");
    delay(3000);
  }
}

```

Result:

Thus the RFID module is interfaced with Arduino and studied.

Exp No:20 SIMULATION USING PROTEUS SOFTWARE –AN INTRODUCTION
Date:

Aim

To simulate a multivibrator using Proteus Software and check its functionality by verifying its output with an simulated LED.

Pre Lab Questions:

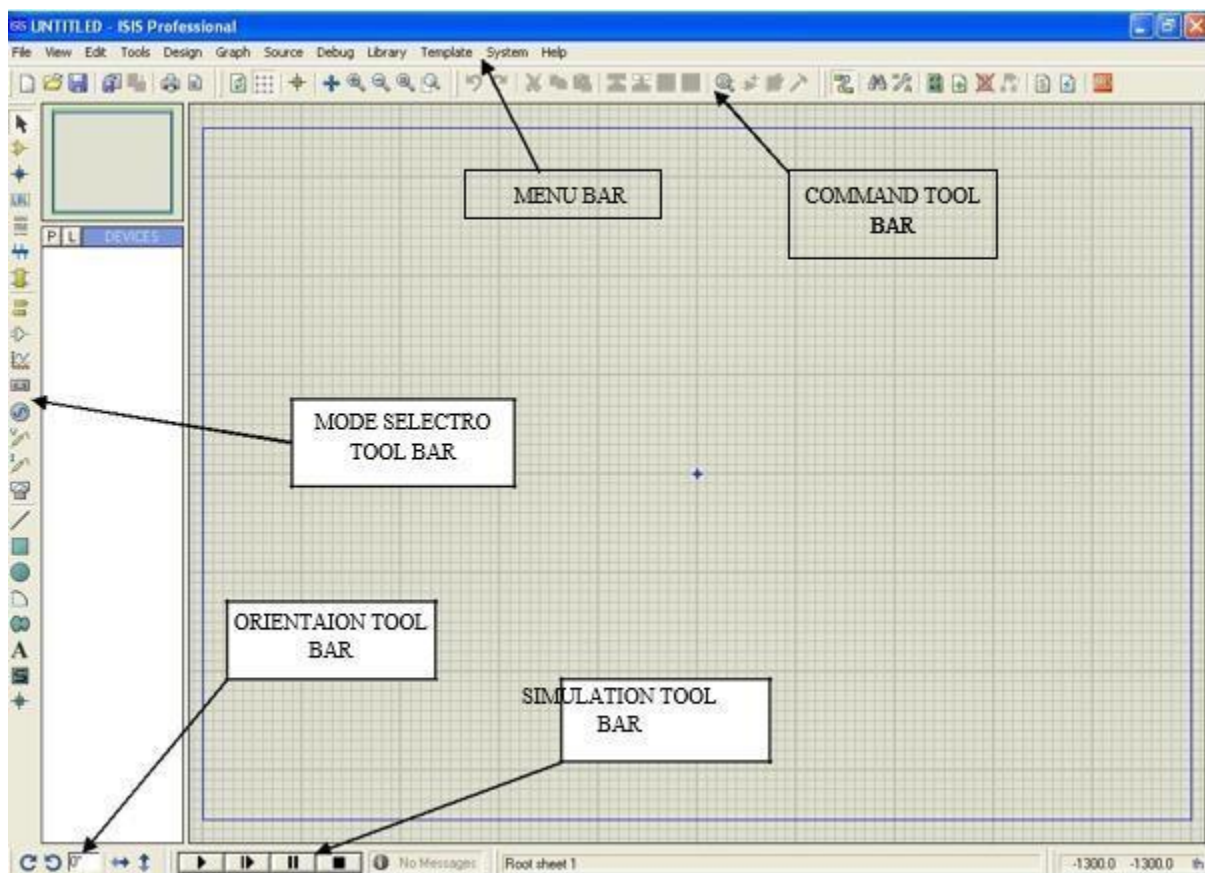
- What is the need for a simulation tool?
- What is Proteus Software?
- List the feature of Proteus Software?
- List some of the design tools in proteus software?
- Give any 3 gadgets available in the proteus software?

Procedure:

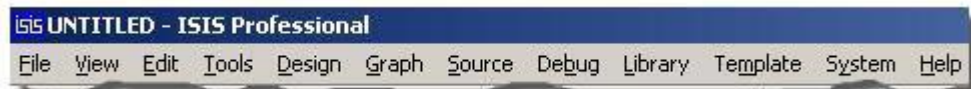
Step1:

Start ➡ All ➡ Programs ➡ Proteus XX Professional
 ➡ ISIS xxProfessional


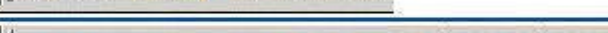

Identify the components






The MenuBar




TheToolbars

TITLE	TOOLBAR
File / Print Commands	
Display Commands	
Editing Commands	
Design Tools	

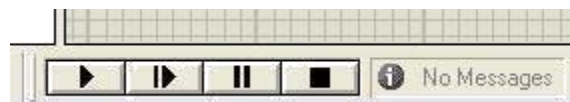
Mode-Selector Toolbar

TITLE	TOOLBAR
Main Modes	
Gadgets	
2D Graphics	

Orientation Toolbar

TITLE	TOOLBAR
Rotation	
Reflection	

Simulation Toolbar



Construct the following multivibrator circuit using analog and digital components and simulate it,

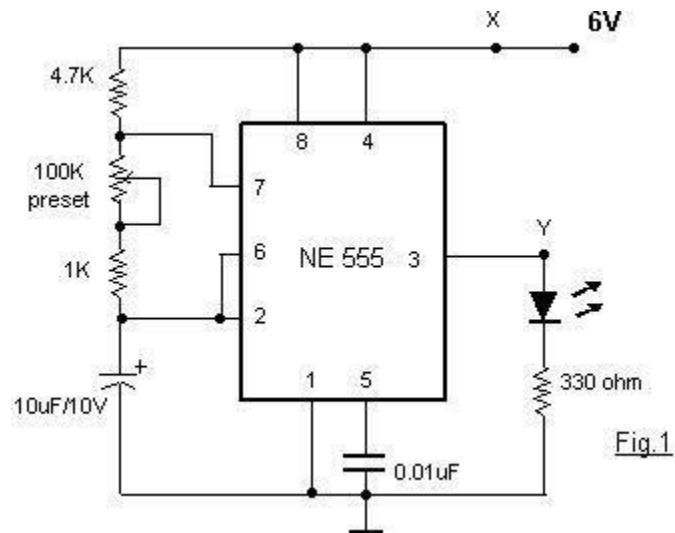


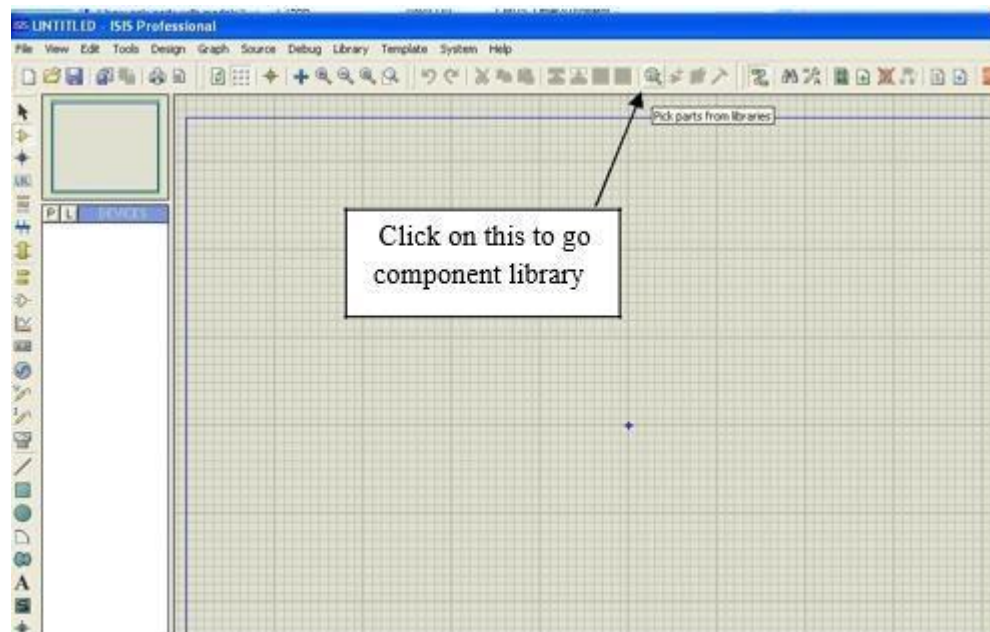
Fig.1

The Components needed are,

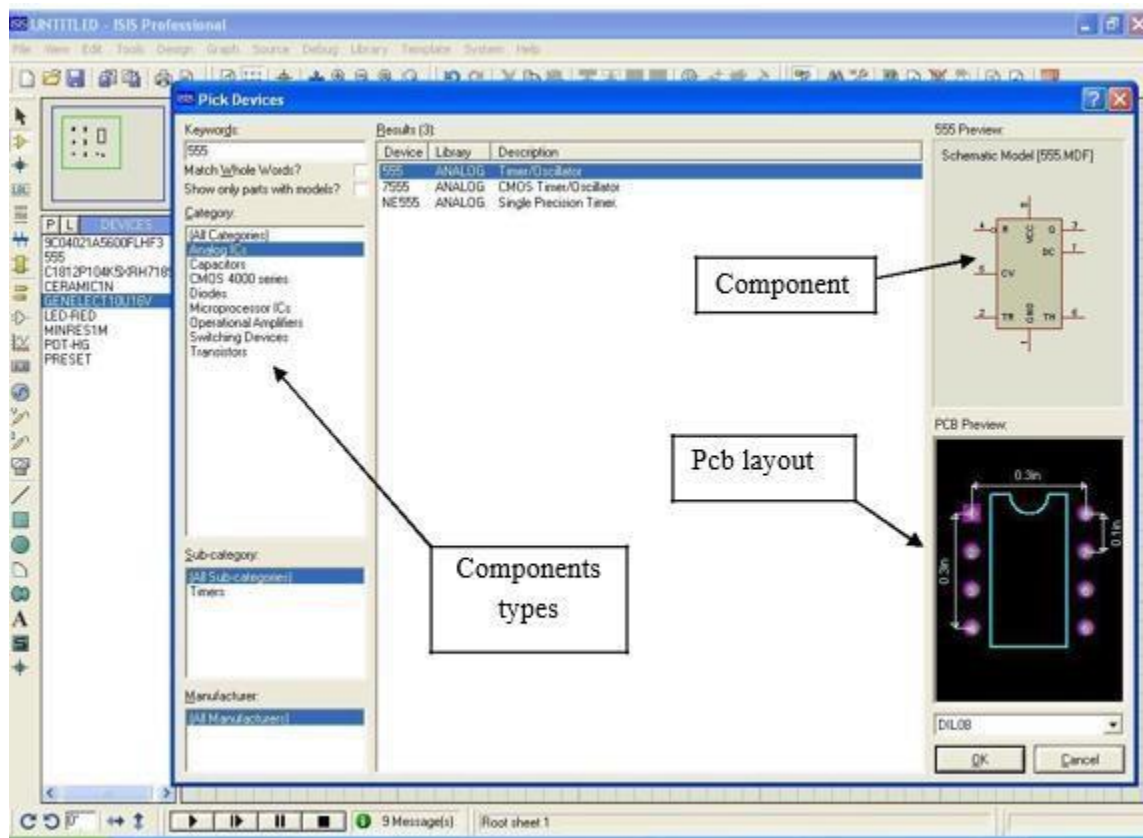
- Resistors 4.7k Ω ,1k Ω 330 Ω , 100k (variable resistor)
- Capacitors 10uF ,0.01uF
- IC NE555
- LED
- Power supply

Step 2:Choose required components from the component library in editing commands.

Editing commands.

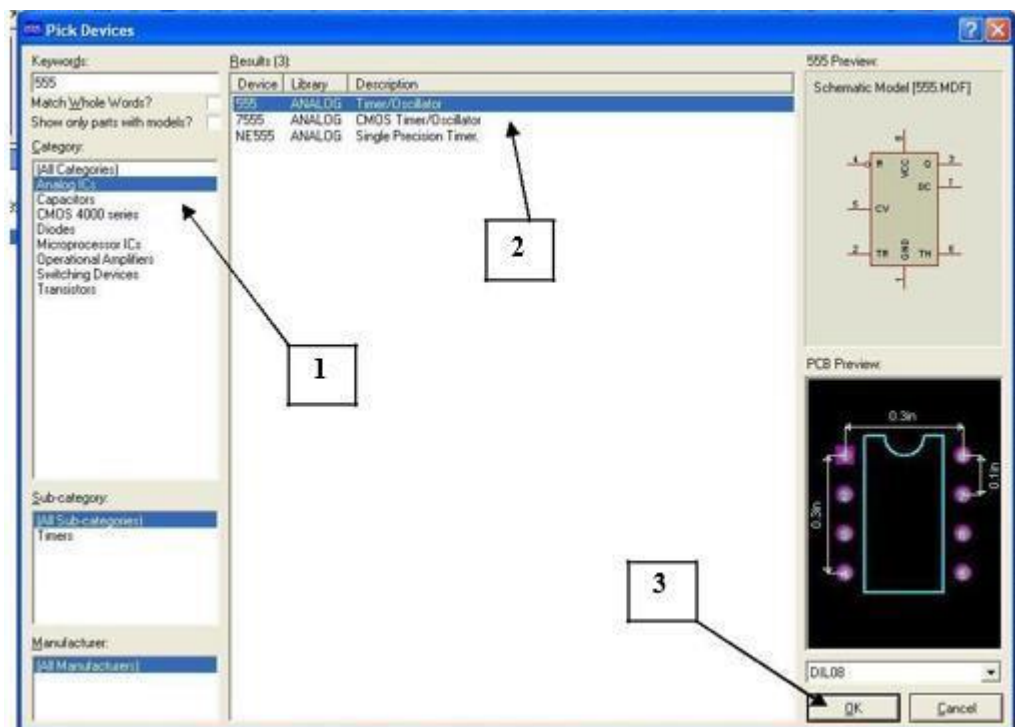


Then the component library window will appear



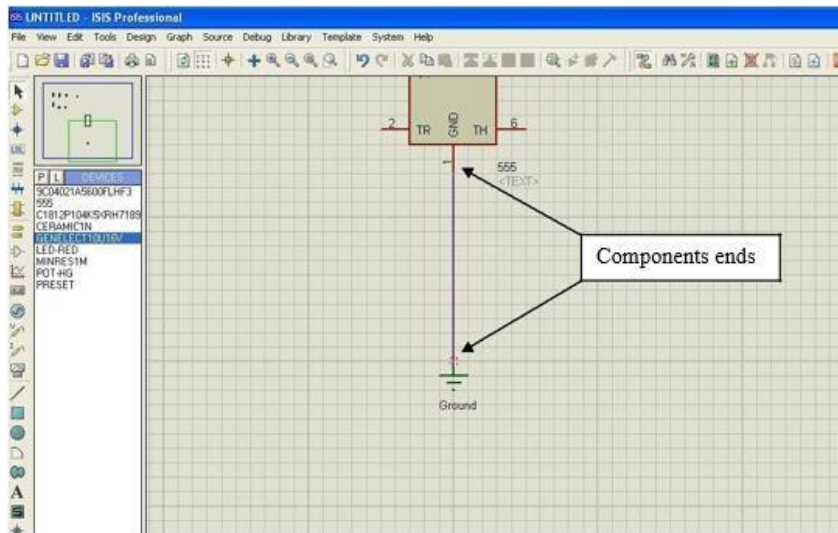
Step 3

Then choose the all require components and put into the main window.



Step 4

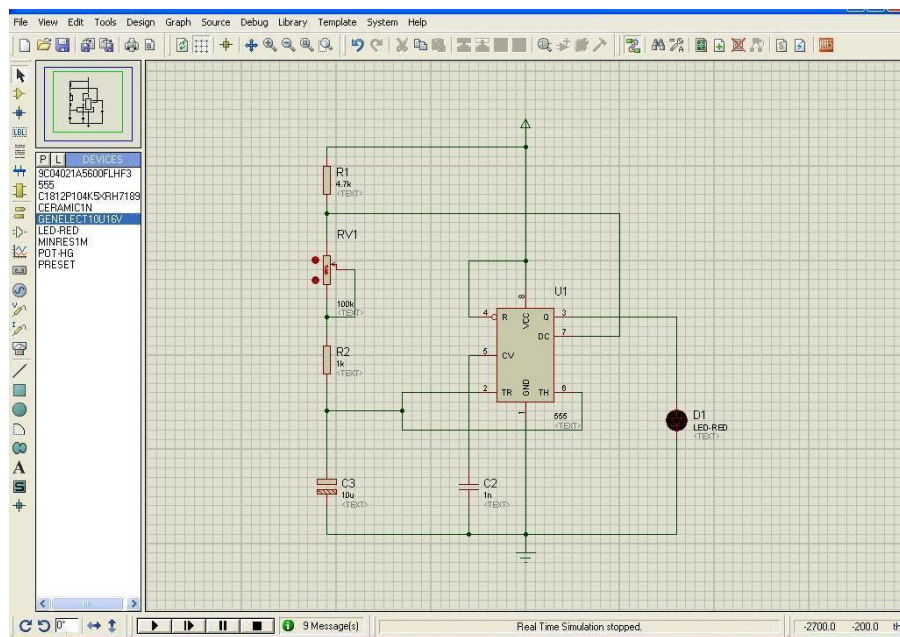
Connect all the components as the above circuit diagram by using connecting lines. We can get connecting lines by selecting ends of the components as shows in following figure.

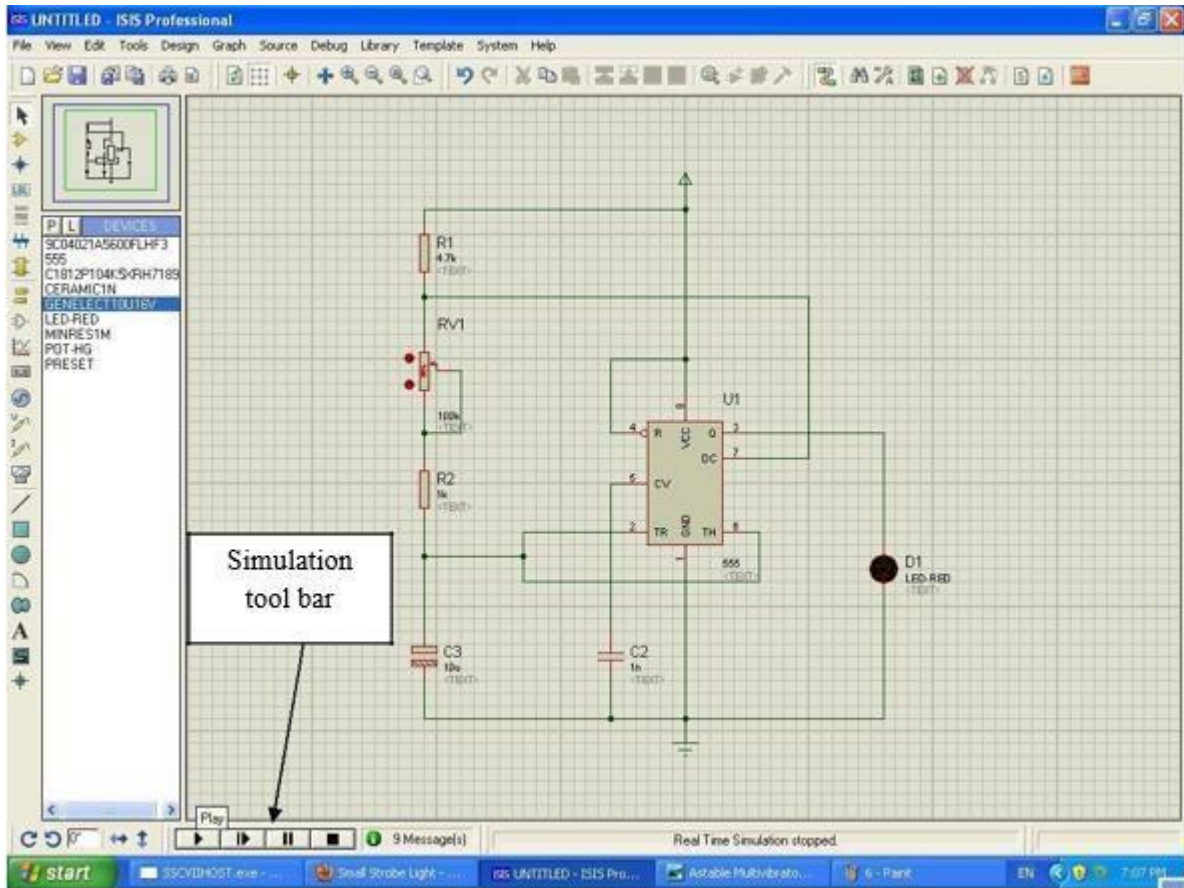


Then complete the circuit using connecting lines as following figure.....

Step 5

Now make sure all the components correctly connected. Then go the simulation tool bar.

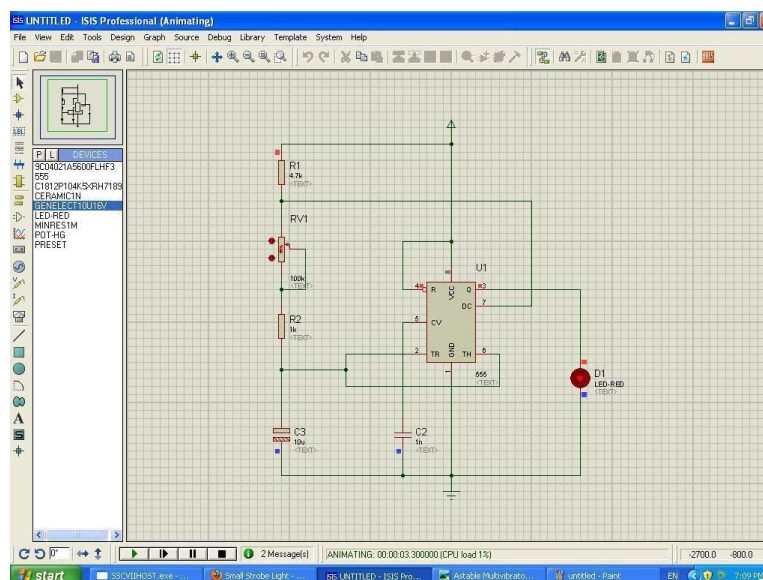




To start the simulation click on play button.

Then start the simulation if there are any errors the simulation must be fail and give error messages.

Check the output by inspecting the status of the LED.



Post Lab Questions:

1. What is mode selector tool bar
2. What is orientation tool bar
3. What is the need for simulation tool bar?
4. What is the frequency of the output obtained in simulation?
5. What is the name of the library under which 555IC is located?

Result:

Thus the Multivibrator is simulated using Proteus Software and the output is Observed.

Exp.No:21 SIMULATION OF CALCULATOR USING 8051 MICROCONTROLLER IN PROTEUS SOFTWARE

Date:

Aim:

To simulate a simple calculator using 8051 microcontroller in Proteus Software.

Pre Lab Questions

1. List some of the microprocessor supported by proteus software
2. List the required arithmetic operations to be performed by the calculator?
3. Write down the features of 8051 microcontroller
4. What is the difference between microprocessor and microcontroller?
5. What is the size of the data bus in 8051 microcontroller?

Procedure:

The calculator we are going to design is quite basic calculator, it will only perform 4 tasks, which are as follows:

When you press the (+) button then it will add the two digits. For example, you want to add 2 and 3 then you need to press 2 + 2 = these four buttons in sequence and when you press the = button it will automatically will give you the sum.

When you press (-) button it will subtract the two digits like 3 – 2 = and it will give you the result.

When you press (x) button it will multiply the two digits.

When you press the (/) button it will simply divide the two digits.

Whenever you press the (=) button, it will give you the output depending on the function you used before and if you press (=) in the start then it will give “Wrong Input”.

Finally, there's (ON/C) button on the Calculator, when you press this it will simply reset the code and will clear the LCD.

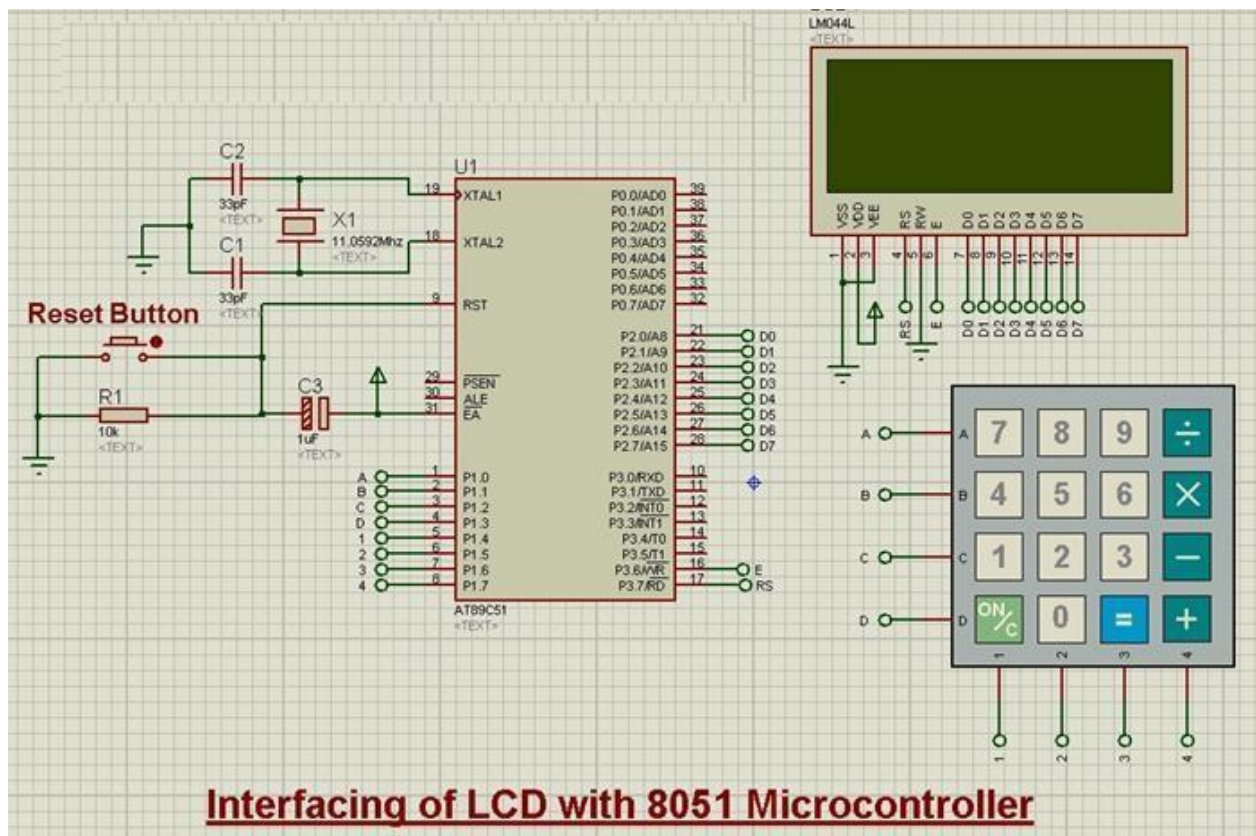
So, that show this calculator is gonna work. More over, it will always reset when you try

As its a simple calculator, so its only limited to 1 digit, means it will only apply the operation on single digit input like $2+3$ but it won't work on more than 1 digit like $12+13$.

After that, we will do the coding part for calculator with 8051 Microcontroller.

So, now let's get started with Proteus Simulation.

The Proteus Simulation of this Calculator with 8051 Microcontroller and is shown in below figure:



Programming code:

while(1)

```
//getnumb1
```

```
key=get_key();
```

```
writecmd(0x01);
```

```

writedata(key);

num1=get_num(key);

//cleardisplay

//Echo the key pressed toLCD

//Get int number from char value, it checksfor
wrong input as well

if(num1!=Error)

//if correct input then proceed,num1==Error
means wrong input

{

//getfunction

key =get_key();

writedata(key);

//Echo the key pressed toLCD

func=get_func(key);

//it checks for wrong func 17

if(func!='e')

//if correct input thenproceed,

func=='e' means wrong input 20

{

//getnumb2

key =get_key();

writedata(key);

//Echo the key pressed toLCD

num2=get_num(key);

```

```

//Get int number fromchar
value, it checks for wrong input as well 26
if(num2!=Error)

//if correct inputthen
proceed, num2==Error means wronginput97
{

//get equal sign key =
get_key();
writedata(key);

//Echo the key pressed to LCD if(key=='=')

//if = is pressedthen
proceed
{

switch(func)

//switch onfunction
{

case '+': disp_num(num1+num2); break; case '-':
disp_num(num1-num2);

break;

case
'x':

disp_num(num1*num2);

break;

case
'/':

```

```

disp_num(num1/num2); break;

}

}

else

46

48here means error wronginput 47

49pressed then clear screen andreset

//key other then=

{

if(key=='C')

write cmd(0x01);

//if clear screen is

//Clear Screen DispError(0);

else

//Display wrong

input error

}

}

}

}

}

}

```

As you can see in the above function, first check for the first key press.

When you pressed the first key on keypad then I get this key and converter it to integer.

After that I waited for the next key which must be some operation Xor/otherwise it will generate the error message.

key like+—

After that code is waiting for the third key which should be some numerical digit and then It converts it to integer again and if you entered some invalid key then it will generate the error.

Finally waiting for the = sign. When you press the = sign it will automatically perform the required operation which I placed in the switch case loop.

It will calculate the value and then print out the result on next key press it will first learn the screen and then get the value and will continue.

Below is the detailed code for the project with comments

```
#include<reg51.h>
```

```
#include<string.h>3
```

```
//DefineMacros
```

```
#defineError
```

```
13
```

```
//Functiondeclarations
```

```
voidcct_init(void);
```

```
voiddelay(int);
```

```
voidlcdinit(void);
```

```
// Any value other than 0 to 9 is good here 6
```

```
and98
```

```
voidwritecmd(int);
```

```
voidwritedata(char);
```

```
voidwriteline(char[]);
```

```
voidReturnHome(void);
```

```
charREAD_SWITCHES(void);
```

```
charget_key(void);
```

```
intget_num(char);
```



```

charget_func(char);

voidDispError(int);

voiddisp_num(int);

void WebsiteLogo();

/

//Pindescription

/*

P2 is databus

P3.7 isRS

P3.6 isE

P1.0 to P1.3 are keypad rowoutputs

P1.4 to P1.7 are keypad column inputs 31

*/

//*****

// DefinePins

//*****

sbit RowA = P1^0;

//RowA

sbit RowB = P1^1;

//RowB

sbit RowC = P1^2;

//RowC

sbit RowD = P1^3;

//RowD

sbit C1

```

```

sbit C2

sbit C3

sbit C4= P1^4;

= P1^5;

= P1^6;

= P1^7;//Column1

//Column2

//Column3

//Column4

sbit E

sbit RS= P3^6;

= P3^7;//E pin for LCD

//RS pin for LCD

//*****

// Mainprogram

//

intmain(void)

{

charkey;

//key char for keeping record ofpressed

key

int num1=0;

//Firstnumber

char func='+';

//Function to be performed amongstwo

```

```

numbers

int num2=0;

//Second number 59

cct_init();

//Make input and output pins asrequired

lcdinit();

//InitilizeLCD

WebsiteLogo();

while(1)

{

WebsiteLogo();

//getnumb1

key =get_key();

writecmd(0x01);

//cleardisplay

WebsiteLogo();

writedata(key);

//Echo the key pressed toLCD

num1=get_num(key);

//Get int number from char value, itchecks

for wrong input as well 73

if(num1!=Error)

//if correct input then proceed,num1==Error

means wrong input 76

{

```

```

//getfunction

key =get_key();

writedata(key);

//Echo the key pressed toLCD

func=get_func(key);

//it checks for wrong func

if(func!='e')

func=='e' means wrong input

{

//get numb2

key = get_key();

writedata(key);

num2=get_num(key);

value, it checks for wrong input as well

//if correct input thenproceed,

//Echo the key pressed toLCD

//Get int number fromchar

if(num2!=Error)

proceed, num2==Error means wrong input

{

//get equal sign

key = get_key();

writedata(key);

LCD

//if correct inputthen

```

```

//Echo the key pressedto
if(key=='=')

//if = is pressedthen
proceed
{
switch(func)

//switch onfunction
{
case '+': disp_num(num1+num2); break;
case '-': disp_num(num1-num2); break;
case 'x': disp_num(num1*num2); break;
case '/': disp_num(num1/num2); break;
}
}

else

//key other then=
here means error wrong input
{
if(key=='C')

//if clear screenis
pressed then clear screen and reset
{

writecmd(0x01);

//Clear Screen
WebsiteLogo();

```

```
}  
  
else  
  
{  
  
DispError(0);  
  
input  
  
//Display  
  
error  
  
WebsiteLogo();  
  
}  
  
}  
  
}  
  
}  
  
}  
  
}  
  
}  
  
}  
  
void WebsiteLogo()  
  
{  
  
writecmd(0x95);  
  
writedata('w');  
  
writedata('w');  
  
writedata('w');  
  
writedata('.');  
  
writedata('T');  
  
writedata('h');  
  
writedata('e');
```

```
writedata('E');
```

```
writedata('n');
```

```
writedata('g');
```

```
writedata('i');
```

```
writedata('n');
```

```
writedata('e');
```

```
writedata('e');
```

```
writedata('r');
```

```
writedata('i');
```

```
writedata('n');
```

```
writedata('g');
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
Wrong100
```

```
writcmd(0xd8);
```

```
writedata('P');
```

```
writedata('r');
```

```
writedata('o');
```

```
writedata('j');
```

```
writedata('e');
```

```
writedata('c');
```

```
writedata('t');
```

```
writedata('s');
```

```
writedata('.');
```

```
writedata('c');
```

```
writedata('o');
```

```
writedata('m');
```

```
writcmd(0x80);
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```

```
//write
```



```

//write

//write

//write

//write

//write

//write

}

voidcct_init(void)

{

P0=0x00;

P1=0xf0;

P2=0x00;

P3=0x00;

}

//notused

//used for generating outputs and taking inputs fromKeypad

//used as data port forLCD

//used for RS andE

void delay(inta)

{

inti;

for(i=0;i<a;i++);

}

//nullstatement

void writedata(chart)

```

```

{
RS=1;

P2=t;

E =1;

delay(150);

E =0;

delay(150);

}

void writecmd(int

{

RS = 0;

P2=z;

E =1;

delay(150);

E =0;

delay(150);

}

// This isdata

//Datatransfer

// => E = 1

// => E = 0

z)

// This is command

//Datatransfer

// => E =1

```

```

// => E =0

void lcdinit(void)

{

////////// Reset process from datasheet //////////

delay(15000);

writecmd(0x30);

delay(4500);

writecmd(0x30);

delay(300);

writecmd(0x30);

delay(650);

////////////////////////////////////////

writecmd(0x38);

//functionset

writecmd(0x0c);

//display on,cursor off,blinkoff

writecmd(0x01);

//cleardisplay

writecmd(0x06);

//entry mode, setincrement

}

voidReturnHome(void)

/* Return to 0 cursor location*/101

{

writecmd(0x02);

```

```

delay(1500);

WebsiteLogo();

}

void writeline(char Line[])

{

int i;

for(i=0;i<strlen(Line);i++)

{

writedata(Line[i]);

}

ReturnHome();

/* Write Character*/

/* Return to 0 cursor position*/

}

char READ_SWITCHES(void)

{

RowA =0; RowB= 1; RowC = 1; RowD= 1;

if (C1

if (C2

if (C3

if (C4==

==

==

==0) {

0) {

```

```

0) {

0) {delay(10000); while

delay(10000); while

delay(10000); while

delay(10000); while(C1==0);

(C2==0);

(C3==0);

(C4==0);

RowA =1; RowB= 0; RowC = 1; RowD= 1;

if (C1

if (C2

if (C3

if (C4==

==

==

==0) {

0) {

0) {

0) {delay(10000); while

delay(10000); while

delay(10000); while

delay(10000); while(C1==0);

(C2==0);

(C3==0);

(C4==0);

```

```
RowA = 1; RowB = 1; RowC = 0; RowD = 1;
```

```
if (C1
```

```
if (C2
```

```
if (C3
```

```
if (C4==
```

```
==
```

```
==
```

```
==0) {
```

```
0) {
```

```
0) {
```

```
0) {delay(10000); while
```

```
delay(10000); while
```

```
delay(10000); while
```

```
delay(10000); while(C1==0);
```

```
(C2==0);
```

```
(C3==0);
```

```
(C4==0);
```

```
RowA = 1; RowB = 1; RowC = 1; RowD = 0;
```

```
if (C1
```

```
if (C2
```

```
if (C3
```

```
if (C4==
```

```
==
```

```
==
```

```
==delay(10000); while
```

```
delay(10000); while
delay(10000); while
delay(10000); while(C1==0);
(C2==0);
(C3==0);
(C4==0);
0) {
0) {
0) {
0) {
return'n';
//Test Row A
return '7'; }
return '8'; }
return '9'; }
return '/'; }
//Test Row B
return '4'; }
return '5'; }
return '6'; }
return 'x'; }
//Test Row C
return '1'; }
return '2'; }
return '3'; }
```

```

return '-'; }

//Test Row D

return 'C'; }

return '0'; }

return '='; }

return '+'; }

// Means no key has beenpressed

}

charget_key(void)

//get key fromuser

{

char key='n';

//assume no key pressed 277

while(key=='n')

//wait untill a key ispressed

key=READ_SWITCHES();

//scan the keys again and again 280

returnkey;

//when key pressed then return its value

}

int get_num(char ch)

{

switch(ch)

{

case '0': return 0; break;

```



```
case '1': return 1; break;

case '2': return 2; break;

case '3': return 3; break;

case '4': return 4; break;

case '5': return 5; break;

case '6': return 6; break;

//convert char into int102
```

Post Lab Questions:

1. What is the need for LCD in the project?
2. What is the need for ADC in the Project?
3. What are the arithmetic operations done in the simulation?
4. What is the need for RESET button in the project?
5. What is the operating frequency of the microcontroller?

Result:

Thus the calculator was simulated using 8051 microcontroller in Proteus Software.