SKILL ACTIVITY NO: 1

Date : 9/10/25

(To be filled by the Instructor)

Title : Basic GPIO control using PIC Microcontroller

Skills / competencies to be acquired :

1. Programming in Embedded C for PIC microcontroller
2. Configuring and controlling GPIO pins in PIC microcontroller
3. Understanding basic ports operations in embedded systems
4. Simulating microcontroller circuits in MPLAB and Proteus

Duration of activity (hours) : 1

(To be filled by the Student)

1. What is the purpose of this activity ? (Explain in 3 - 4 lines)

The purpose of this activity is to introduce students to basic GPIO control using a PIC microcontroller. By writing a simple program to toggle an output pin, students will learn how to configure ports, set output values, and understand bit manipulation in embedded systems

2. Steps performed in this activity (Explain in 5 - 6 lines)

① Create a new project in MPLAB
② Write the embedded C code
③ Build and compile the code
④ Create a new simulation project in Proteus
⑤ Add components to the the schematic
⑥ Load the HEX file
⑦ Run the simulation

3. What resources / materials / equipments / tools did you use for this activity ?

1. MPLAB X IDE
2. Proteus software
3. PIC microcontroller
4. LCD and resistor
5.
6.
7.
8.

4. What skills did you acquire ?

1. Basic configuration of PIC microcontroller
2. Understanding bitwise operations in Embedded C
3. Implementing simple output control
4. using GPIO ports
5. Simulating and testing embedded systems in Proteus
6. Generating Hex file
7.
8.

5. Time taken to complete the activity ? 1 (hours)

(Signature)
Instructor

(Signature)
Student

SKILL ACTIVITY NO: 2

(To be filled by the Instructor)

Date : 23/11/25

Title : LED chasing pattern using PIC microcontroller

Skills / competencies to be acquired :

1. Programming in Embedded c for PIC
2. microcontroller
3. Configuring and controlling GPIO pins
4. in PIC microcontroller
5. Simulating microcontroller circuits
6. in MPLAB and Proteus
7. Understanding LED sequencing and
8. delay functions

Duration of activity (hours) : 9

(To be filled by the Student)

1. What is the purpose of this activity ? (Explain in 3 - 4 lines)

The purpose of this activity is to develop an understanding of microcontroller based LED control and sequencing. By implementing a simple LED chasing pattern using a PIC microcontroller, students will gain hands on experience in embedded systems programming and circuit simulation

2. Steps performed in this activity (Explain in 5 - 6 lines)

① Create a new project in MPLAB (PIC 16F877A)
② Write the embedded C code
③ Build and compile the code
④ Create a new simulation project in Proteus
⑤ Add components to schematic
⑥ Load the HEX file
⑦ Run the simulation

3. What resources / materials / equipments / tools did you use for this activity ?

1. MPLAB IDE
2. Proteus software
3. PIC microcontroller
4. LEDs and resistor
5. _____
6. _____
7. _____
8. _____

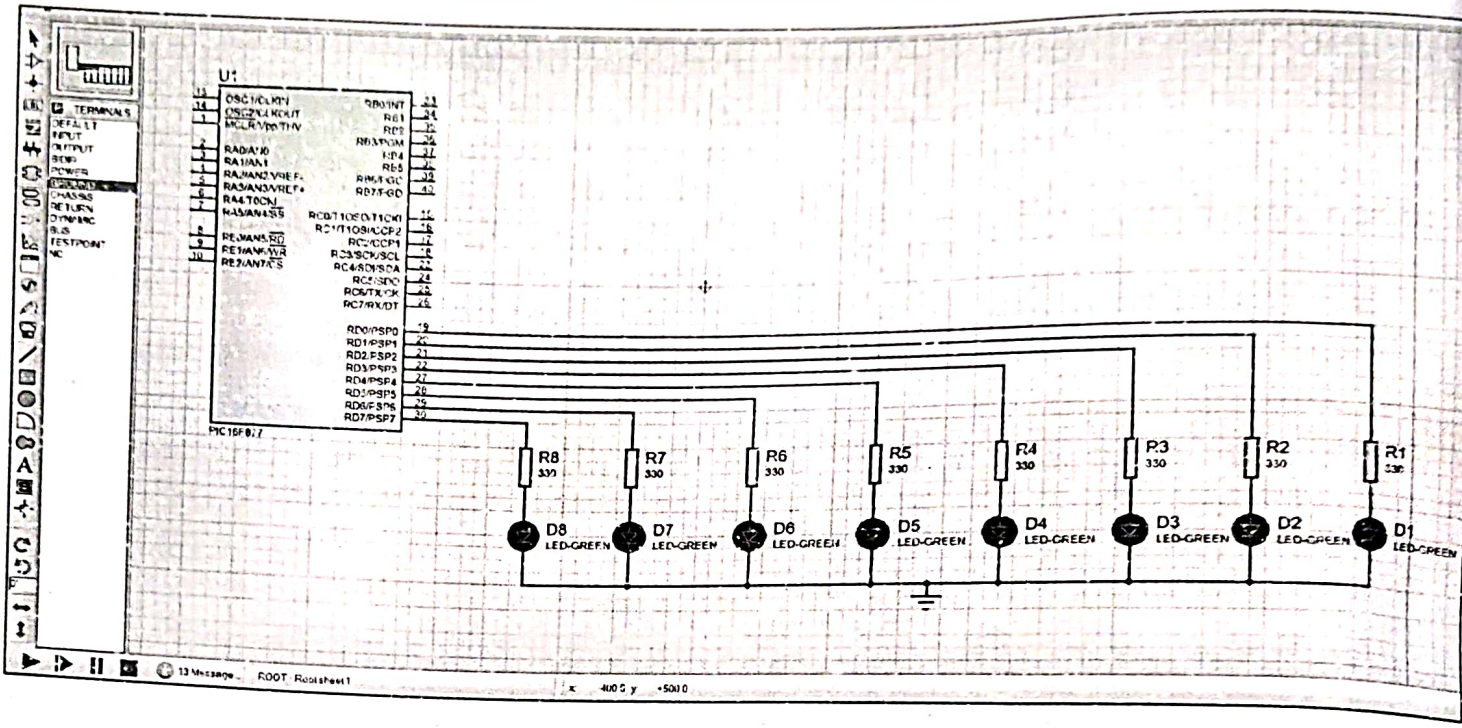
4. What skills did you acquire ?

1. Configuring and programming PIC
2. micro controller GPIOs
3. implementing delay functions in
4. embedded C
5. Creating LED chasing patterns
6. using bitwise operations
7. simulating microcontroller in
8. Proteus

5. Time taken to complete the activity ? 1 (hours)

(Signature)
Instructor

(Signature)




```c
// Program to implement a LED pattern where a set of LEDs remain ON while others blink one at a time

#include <xc.h>  //Includes compiler file into the program during execution
void msdelay (unsigned int time);
void main()
{
    TRISD= 0x00;    // Set PORTD as output
    unsigned char constant;
    while(1)
    {
        for (int i = 7; i >=0 ; i--)
        {
            constant = 0xFF >> i; // Selects LEDs to stay on
            for(int j=7-i;j<=7;j++)
            {
                PORTD=0x01<<j|constant; // Turns on LEDs other than constant one at a time
                msdelay(150);
                PORTD = constant; // Sets PORTD to keep the constant LEDs on
                msdeley(150);
            }
        }
    }
}
void msdelay (unsigned int time)
{
    unsigned int i,j;
    for (i=0; i< time; i++)
    for(j=0; j<710; j++);
}
```

SKILL ACTIVITY NO: 3

Date : 6/2/25

(To be filled by the Instructor)

Title : Design and simulation of digital voltmeter using PIC 18 microcontroller

Skills / competencies to be acquired :

1. Understanding ADC using PIC
2. Programming in embedded C with
3. MPLAB XC8
4. Interfacing 16 x2 LCD with PIC 18
5. Simulating circuits in Proteus
6. for real time testing
7. Implementing precise voltage
8. measurement using microcontroller based ADC

Duration of activity (hours) : 1

(To be filled by the Student)

1. What is the purpose of this activity ? (Explain in 3 - 4 lines)

The purpose of this activity is to design, program, and simulate a digital voltmeter using a PIC 18F microcontroller. This project allows students to understand ADC operations, practice embedded C programming, and develop practical skills in circuit simulation using Proteus. By completing this activity, students will gain experience in real time voltage measurement and display techiques

2. Steps performed in this activity (Explain in 5 - 6 lines)

① Setting up the MPLAB project (PIC 18F 4550)
② Writing and compiling code in MPLAB
③ Circuit design in Proteus
④ Running the simulation in project
⑤ Check the Hex file
⑥ Upload it onto microcontroller

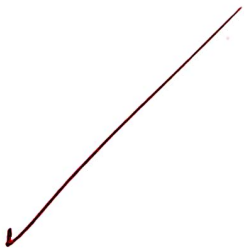
3. What resources / materials / equipments / tools did you use for this activity ?

1. MP LAB x IDE
2. XC8 compiler
3. Proteus 8
4.
5.
6.
7.
8.

4. What skills did you acquire ?

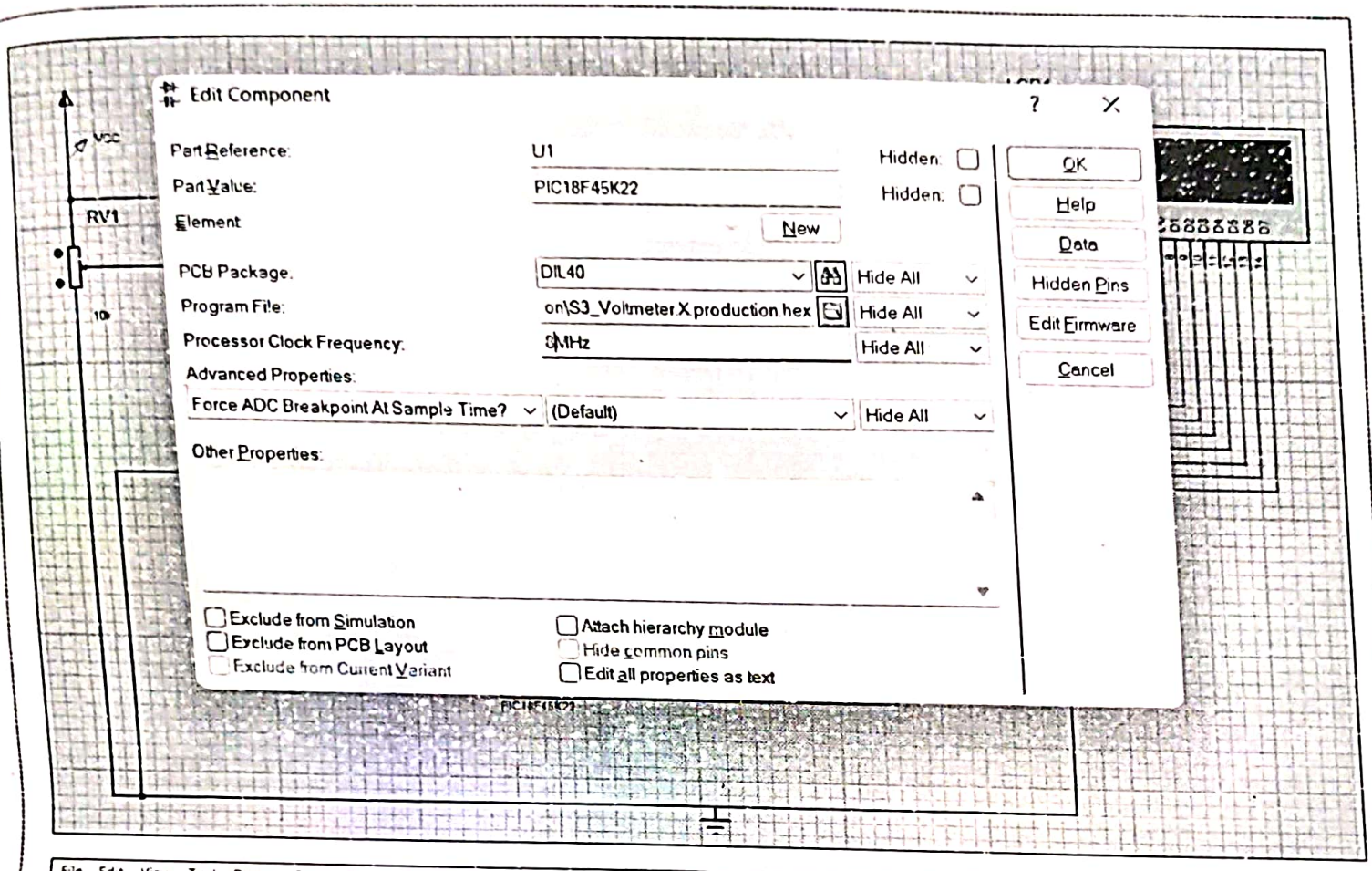
1. Embedded systems development using
2. MPLAB and Proteus
3. Microcontroller ADC Interfacing for
4. voltage measurement
5. Circuit design and simulation
6. in Proteus
7. debugging and testing embedded
8. applications

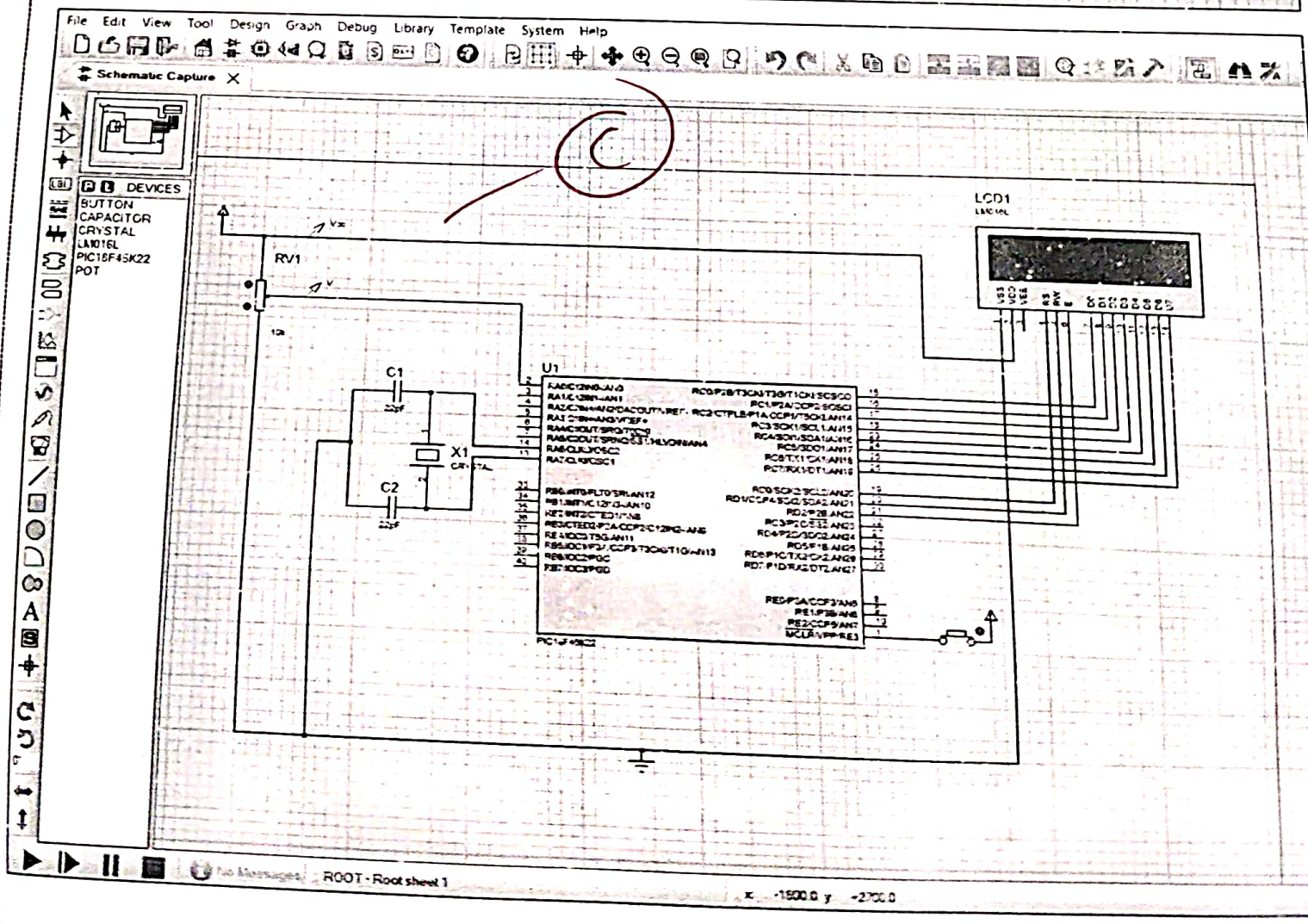
5. Time taken to complete the activity ? 1 (hours)



(Signature)
Instructor

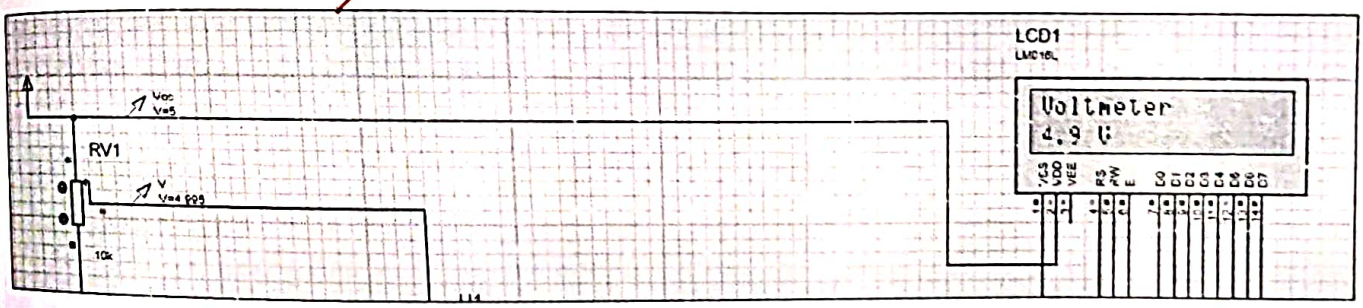
(Signature)
Student

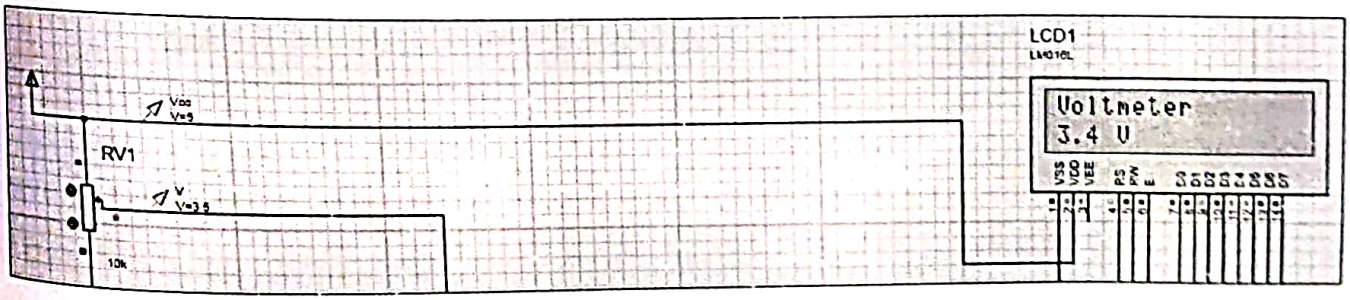

Edit Component
Part Reference: U1
Hidden:
Part Value: PIC18F45K22
Hidden:
Element
New
PCB Package: DIL40
Hide All
Program File: on\S3_Voltmeter X production hex
Hide All
Processor Clock Frequency: 8MHz
Hide All
Advanced Properties:
Force ADC Breakpoint At Sample Time?
(Default)
Hide All
Other Properties:
Exclude from Simulation
Exclude from PCB Layout
Exclude from Current Variant
Attach hierarchy module
Hide common pins
Edit all properties as text
OK
Help
Data
Hidden Pins
Edit Firmware
Cancel
RV1
VCC


File Edit View Tool Design Graph Debug Library Template System Help
Schematic Capture
DEVICES
BUTTON
CAPACITOR
CRYSTAL
LM016L
PIC18F45K22
POT
RV1
C1
C2
X1
U1
LCD1
ROOT - Root sheet 1

```c
/*
 * File:   LCD_meter.c
 *
 */


#include <xc.h> //Header file of compiler
#include <stdint.h>     //Defines integer types, their limits, and macros for integer constants

#pragma config MCLRE= EXTMCLR, WDTEN=OFF, FOSC=HSHP
#define _XTAL_FREQ 8000000  //Sets oscillator frequency

//LATx register used to hold last value written to register pin
#define RS LATDbits.LATD0
#define RW LATDbits.LATD1
#define EN LATDbits.LATD2

void LCD_Command(uint8_t cmd);
void LCD_Data(uint8_t cmd);
void WriteToLCD(uint8_t* p, uint8_t pos);

void main(void) {
    uint16_t result;                    //ADC result variable
    uint8_t text[16];                   //array for string to send to LCD

    ANSELAbits.ANSA0 = 1;               //digital input buffer disabled
    TRISAbits.RA0 = 1;                  //RA0 configured as an input
    TRISC = 0;                          //PORTC configured as an output
    TRISD = 0;                          //PORTD configured as an output

    //ADC converter set up
    ADCON0bits.CHS = 0b00000;           //AN0 channel selected
    ADCON0bits.ADON = 1;                //ADC enabled
    ADCON1bits.PVCFG = 0b00;            //Vref+ connected to Vdd
    ADCON1bits.NVCFG = 0b00;            //Vref- connected to Vss
    ADCON2bits.ADFM = 1;                //result right justified
    ADCON2bits.ACQT = 0b100;            //acquisition time = 8 TAD
    ADCON2bits.ADCS = 0b011;            //TAD = 1.7 us (FRC))

    ADCON0bits.GO = 1;                  //ADC conversion starts

    EN = 0;
    __delay_ms(100);
    LCD_Command(0x38);                  //LCD 2 lines, 5x7 matrix
    __delay_ms(100);
    LCD_Command(0x0C);                  //display on, cursor off
    __delay_ms(5);

    WriteToLCD((uint8_t*)"Voltmeter", 0x80);
    while(1)
    {
        while(ADCON0bits.nDONE);
        result = ((ADRESL | (uint16_t)ADRESH << 8) * 50) >> 10;
        uint8_t* p = text;
        *p++ = 0x30 |(uint8_t)(result/10); //one digit before comma
        *p++ = 0x2E;
        *p++ = 0x30 |(uint8_t)(result%10); //one digit after comma
        *p++ = 0x20;
        *p = 0x56;
        WriteToLCD((uint8_t*)text, 0xC0);  //display result

        ADCON0bits.GO = 1;
    }
    return;
```

```
}
void WriteToLCD(uint8_t* p, uint8_t pos)
{
    LCD_Command(pos);
    __delay_ms(5);
    while(*p)
    {
        LCD_Data((uint8_t)*p++);
        __delay_ms(5);

    }
}
void LCD_Command(uint8_t cmd)
{
    LATC = cmd;
    RS = 0;
    RW = 0;
    EN = 1;
    __delay_ms(1);
    EN = 0;
}
void LCD_Data(uint8_t cmd)
{
    LATC = cmd;
    RS = 1;
    RW = 0;
    EN = 1;
    __delay_ms(1);
    EN = 0;
}
```





SKILL ACTIVITY NO: 4

Date : 26/2/25

(To be filled by the Instructor)

Title : Playing Happy Birthday Tune using PIC microcontroller

Skills / competencies to be acquired :

1. Programming in embedded c
2. configuring and using timers for
3. frequency generation
4. Implementing interrupt based sound
5. generation
6. Understanding musical note
7. frequencies and durations
8.

Duration of activity (hours) : 1

(To be filled by the Student)

1. What is the purpose of this activity ? (Explain in 3 - 4 lines)

The purpose of this activity is to develop an understanding of sound generation using PIC microcontroller. By implementing an simple (Happy birthday) using and embedded systems, students will gain hands on experience with timers, interrupts and frequency calculations.

2. Steps performed in this activity (Explain in 5 - 6 lines)

① Create a new project in MPLAB Cog PIC 16F887A)
② Write the embedded C code
③ Build and compile the code
④ Create a new simulation project in Proteus
⑤ Add components to schematic
⑥ Load HEX file
⑦ Run the simulation

3. What resources / materials / equipments / tools did you use for this activity ?

1. MPLAB IDE
2. Proteus software
3. PIC microcontroller
4. Timer and interrupt configuration
5.
6.
7.
8.

4. What skills did you acquire ?

1. Configuring and programming PIC
2. microcontroller timers
3. Implementing sound generation
4. using PIC micro controller
5. understanding musical notes
6. and frequency calculations
7. Simulating and debugging
8. microcontroller circuits in Proteus

5. Time taken to complete the activity ? 1 (hours)

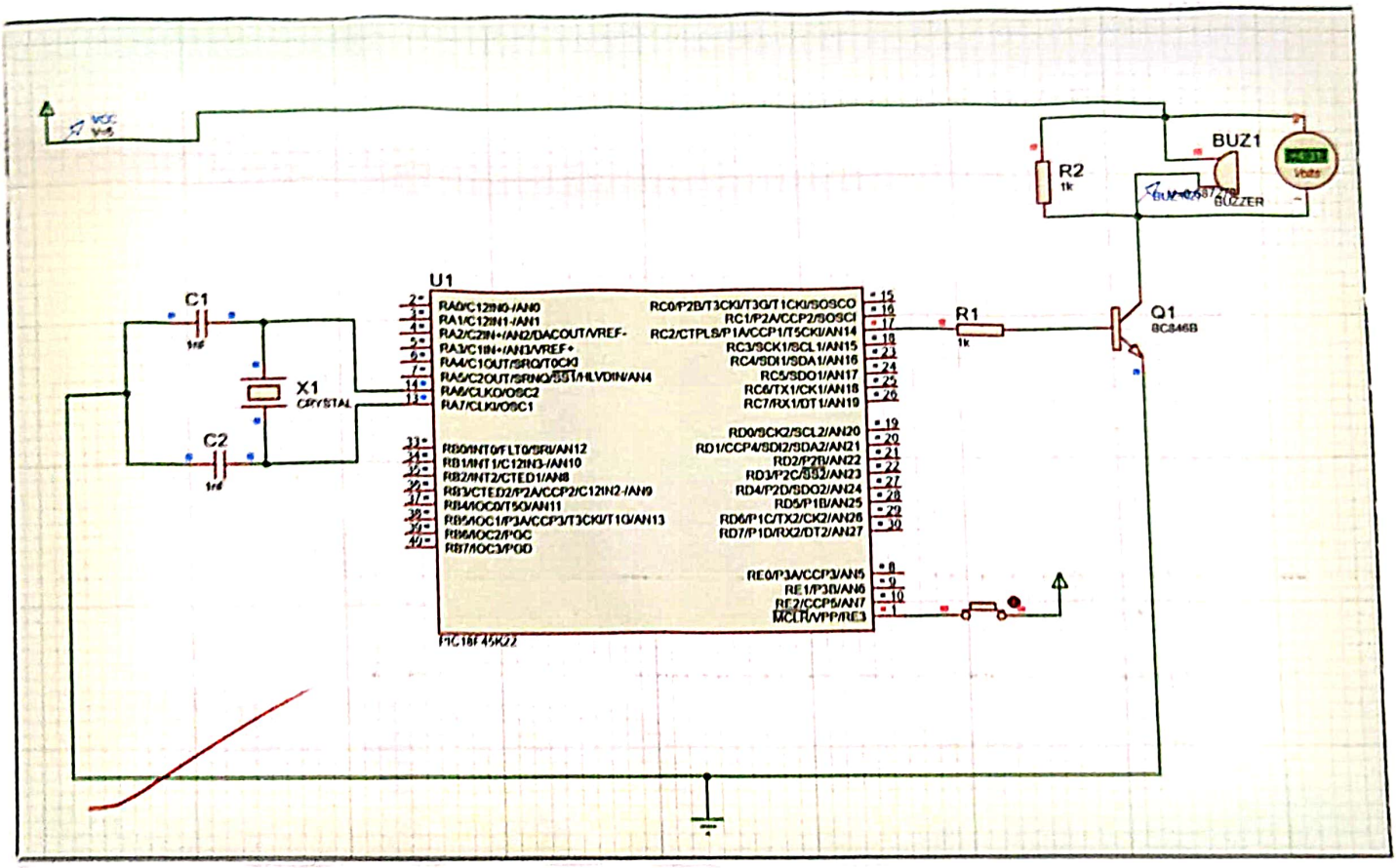
(Signature)
Instructor

(Signature)
Student

Code

```
#include <xc.h>
#include <stdint.h>
#pragma config MCLRE= EXTMCLR, WDTEN=OFF, FOSC=HSHP
#define _XTAL_FREQ 8000000

void __interrupt(low_priority) myLoIsr(void);
void delay_ms(uint16_t time);

uint16_t notesFrequency[] ={262,277,294,311,330,349,370,392,415,440,466,494,524};  //frequency of notes
const uint8_t notesSequence[] ={0,0,2,0,5,4,0,0,2,0,7,5,0,0,12,9,5,4,2,10,10,9,5,7,5};   //sequence to get happy birthday
const uint8_t notesDuration[] ={1,1,2,2,2,3,1,1,2,2,2,3,1,1,2,2,2,2,2,1,1,2,2,2,4};      //notes duration in milliseconde

uint16_t timer;                         //global variable to store the timer for the current note

void main(void) {

    TRISCbits.RC2 = 0;

    //set up timer
    INTCONbits.GIE = 1;                 //global interrupt enabled
    INTCONbits.TMR0IE = 1;              //timer0 interrupt enabled
    INTCONbits.TMR0IF = 0;              //timer0 flag reset (caution)

    T0CONbits.T08BIT = 0;               //16-bit timer selected
    T0CONbits.T0CS = 0;                 //counter option disabled
    T0CONbits.PSA = 0;                  //prescaler used
    T0CONbits.T0PS = 0b000;             //prescaler = 2

    for(int i=0; i<13; i++)
        notesFrequency[i] = (65536 - ((50000/notesFrequency[i])*10));      //timer input calculation

    for(uint8_t i=0; i<25; i++)
    {
        timer = notesFrequency[notesSequence[i]];          //timer is set with frequency of the note to be played
        TMR0 = timer;                                      //timer is loaded
        T0CONbits.TMR0ON = 1;                              //timer is started
        delay_ms(notesDuration[i]*400);                    //delay corresponding to the note duration
        T0CONbits.TMR0ON = 0;                              //timer is stopped before moving to the next note
    }
}

void __interrupt(low_priority) myLoIsr(void)
{
    T0CONbits.TMR0ON = 0;
    LATCbits.LATC2 = ~LATCbits.LATC2;
    TMR0 = timer;
    INTCONbits.TMR0IF = 0;
    T0CONbits.TMR0ON = 1;
}


void delay_ms(uint16_t time)
{
    for(uint16_t i=0; i<time; i++)
        __delay_ms(1);
}
```

output


VCC
R2
1k
BUZ1
BUZZER
C1
1nf
X1
CRYSTAL
C2
1nf
U1
RA0/C12IN0-/AN0
RA1/C12IN1-/AN1
RA2/C2IN+/AN2/DACOUT/VREF-
RA3/C1IN+/AN3/VREF+
RA4/C1OUT/SRQ/T0CKI
RA5/C2OUT/SRNQ/SS1/HLVDIN/AN4
RA6/CLKO/OSC2
RA7/CLKI/OSC1
RB0/INT0/FLT0/SRI/AN12
RB1/INT1/C12IN3-/AN10
RB2/INT2/CTED1/AN8
RB3/CTED2/P2A/CCP2/C12IN2-/AN9
RB4/KBI0/T5G/AN11
RB5/KBI1/P3A/CCP3/T3CKI/T1G/AN13
RB6/KBI2/PGC
RB7/KBI3/PGD
RC0/P2B/T3CKI/T3G/T1CKI/SOSCO
RC1/P2A/CCP2/SOSCI
RC2/CTPLS/P1A/CCP1/T5CKI/AN14
RC3/SCK1/SCL1/AN15
RC4/SDI1/SDA1/AN16
RC5/SDO1/AN17
RC6/TX1/CK1/AN18
RC7/RX1/DT1/AN19
RD0/SCK2/SCL2/AN20
RD1/CCP4/SDI2/SDA2/AN21
RD2/P2D/AN22
RD3/P2C/SS2/AN23
RD4/P2D/SDO2/AN24
RD5/P1B/AN25
RD6/P1C/TX2/CK2/AN26
RD7/P1D/RX2/DT2/AN27
RE0/P3A/CCP3/AN5
RE1/P3B/AN6
RE2/CCP5/AN7
MCLR/VPP/RE3
PIC18F45K22
R1
1k
Q1
BC846B

C

SKILL ACTIVITY NO: 5

Date : _____

(To be filled by the Instructor)

Title : To study sine wave generator using ARM Cortex M3

Skills / competencies to be acquired :

1. understood ARM features
2. understand sine wave generator
3. Calculate constant K for given
4. resonator frequency
5. _____
6. _____
7. _____
8. _____

Duration of activity (hours) : 1

(To be filled by the Student)

1. What is the purpose of this activity ? (Explain in 3 - 4 lines)

The purpose of this activity is to understand how to generate sine wave using ARM cortex m3 microcontroller. It involves configuring the code, adjusting frequencies and amplitude parameters and observing output waveform. This helps in learning ARM features and signal generation techniques

2. Steps performed in this activity (Explain in 5 - 6 lines)

1) Connect the cortex module and LCD to the ARM cortex m3 using connection diagram given in LW
2) Open project sine.cdb on coocox which contains the program
3) In the main.c file change the value of constant K acc to the sampling and resonator frequency and also amplitude.
4) Complete the programs and load .hex file into ARM cortex using coflash
5) Connect the codec o/p to oscilloscope and observe waveforms

3. What resources / materials / equipments / tools did you use for this activity ?

1. ARM Cortex M3 module
2. Elo train
3. I^2C codec module
4. I^2C LED
5. LAI virtual instruments and
6. manual
7.
8.

4. What skills did you acquire ?

1. Understand ARM features
2. Understood sine wave generator
3. Calculate constant K for given
4. resonator frequency
5.
6.
7.
8.

5. Time taken to complete the activity ? 1 (hours)

(Signature)
Instructor

(Signature)
Student

Sine waveform function represented in Taylors series form

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} \Rightarrow x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

To implement this function in a micro controller the z transform of the above equation is used as Taylor series takes a lot of computing power and time

Eqn used $y(n) = K \cdot y(n-1) - y(n-2)$ where $K = 2 \cos\left[2\pi \frac{fo}{Fs}\right]$

fo = Resonator frequency

Fs = sampling frequency

Value of amp for different resonator frequencies

Frequency (in Hz)	Value of amp
1 <	0x008 <
100	0x800
300	0x1600
500	0x2400
700	0x3200
1300	0x6400

Value of K and amp for Fs = 8KHz and Fo = 100 Hz

$$K = 2 \cos\left[2\pi \frac{100}{8 \times 10^3}\right]$$

$$K = 1.9938\,34667$$

$\therefore$ amp = 0x800

Observations

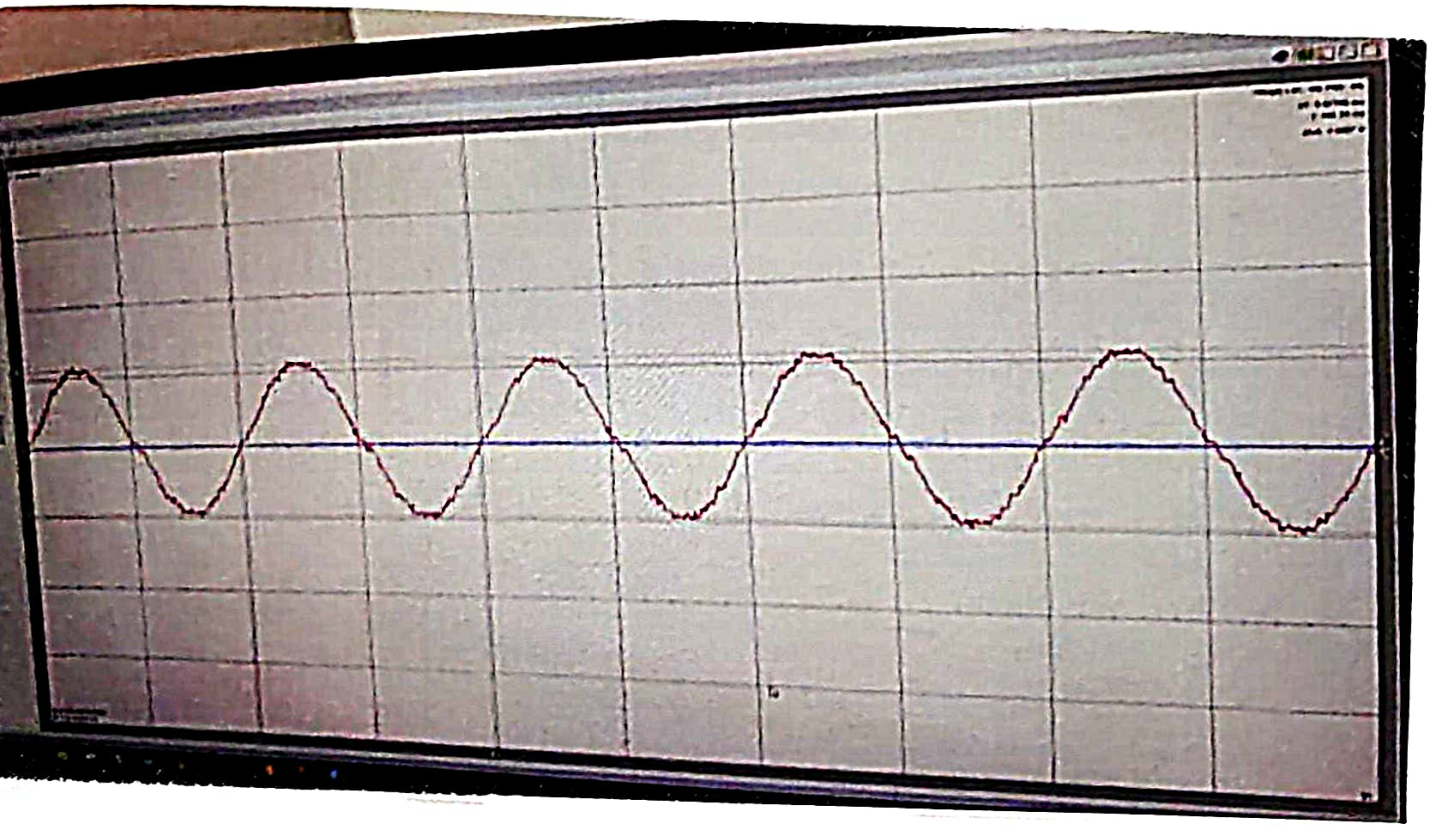
Theoretical F = 100 Hz

Practical

F = 100 · 23 Hz

V = 1 · 03 V

Time period = 9 · 97 ms



main.c

```
#include "audio.h"
#include "delay.h"
#include "lcd.h"

unsigned int i,In;
static double sine_data[3];

void init_sine(void)
{
    sine_data[0]=-1;
    sine_data[1]=0;
    sine_data[2]=1;
}

// sine function ----------------------------------------------------------
short int sine(void)
{
    const float k = 1.9938346; //1.9999944483;      // coeffizient for sine wave: k = 2
 * cos(2 * PI * f0 / fs)
    const short int amp = 0x800;//0x40;
    short int out;
    sine_data[0] = sine_data[1] * k - sine_data[2];
    sine_data[2] = sine_data[1];
    sine_data[1] = sine_data[0];
    out = (short int)sine_data[0] * amp;
    return out;
}

// filter function for IRQ_Handler
void sine_filter(int16_t* l_ch, int16_t* r_ch) {
    *l_ch = sine();
    *r_ch = 0;
}
/**************************************************************************//**
 * @brief       main function
 * @param[in]   none
 * @return      none
 *****************************************************************************/
int main(void)
{
    LCD_INIT();                             // LCD init
    LCD_gotoXY(1,0);                        // column 1, char 0
    LCD_SENDSTRING("    SINE");                 // send text to LCD
    LCD_gotoXY(2,0);                        // column 1, char 0
    LCD_SENDSTRING("  GENERATOR");                  // send text to LCD
    delay_ms(1);                            // wait 1 ms
    init_sine();                            // init the sine generator
    audio_I2C_init();                       // init I2C interface for codec
    audio_init();                           // init audio interface
    audio_set_filter(sine_filter);          // set sine function for signal processing

    while(1)                                // endless loop
    {
        delay_ms(200);                      // wait 200 ms
    }
}
```

SKILL ACTIVITY NO: 6

Date : _____

(To be filled by the Instructor)

Title : To study square wave generator using ARM Cortex m3

Skills / competencies to be acquired :

1. Understand ARM features
2. Understand square wave generator
3. understand fourier series of
4. square wave generator
5. _____
6. _____
7. _____
8. _____

Duration of activity (hours) : 1

(To be filled by the Student)

1. What is the purpose of this activity ? (Explain in 3 - 4 lines)

The purpose of this activity is to understand to generate a square wave using ARM Cortex m3 . It involves configuring the code, adjusting frequency and amplitude parameters and observing the output waveform . This helps in learning ARM features and signal generation techniques

2. Steps performed in this activity (Explain in 5 - 6 lines)

1) Connect the Codec module and LED to ARM cortex m3 using LN connection
2) open the project square . cob on CooCox IDE which contains square generating program
3) In the program change value of K according to the values of resonator sampling and harmonic frequencies
4) Compile the program and load . hex into cortex using coFlash
5) Connect the Codec o/p to the LN oscilloscope and observe the waveform

3. What resources / materials / equipments / tools did you use for this activity ?

1. ARM cortex M3 module
2. Elo train
3. I^2C codec module
4. I^2C LCD
5. LN virtual instruments
6.
7.
8.

4. What skills did you acquire ?

1. Understood ARM features
2. Understood square wave generator
3. Understood fourier series for
4. square wave for generation
5.
6.
7.
8.

5. Time taken to complete the activity ? 1 (hours)

(Signature)
Instructor

(Signature)
Student

on the basics of Fourier analysis a square wave consists of a fundamental sine wave (of the same frequency as the square wave) and odd harmonics of the fundamental.

Fourier series for square $\Rightarrow F(x) = \frac{4}{\pi} \sum_{n=1,3,5\ldots}^{\infty} \frac{1}{n} \sin\left(\frac{n\pi x}{L}\right)$

$\therefore$ Eqn used out $= \frac{4}{\pi} \left(y_1(n) + y_3(n) + y_5(n) + \ldots \right)$

where $y_0[n] = K \cdot y_a[n-1] - y_a[n-2]$

where a is a odd number

$$K = 2 \cos\left[2\pi \frac{F_0}{F_S}\right]$$

Value of K and amp

Fundamental $F = F_0 = 100\,Hz$

Sampling $F = F_S = 8\,KHz$

odd harmonics $= f_1 = 300\,Hz$

$f_2 = 500\,Hz$

amp $= 0 \times 800$

$K_0 = 2 \cos\left[2\pi \frac{100}{8000}\right] \Rightarrow K_0 = 1.993834667$

$K_1 = 2 \cos\left[2\pi \frac{300}{8000}\right] \Rightarrow K_1 = 1.994739841$

$K_2 = 2 \cos\left[2\pi \frac{500}{8000}\right] \Rightarrow K_2 = 1.847759065$

Observation

Theoretical $\Rightarrow$ 100 HZ

Practical $\Rightarrow$ F = 100.23 HZ

V = 1.30263 V

```
/*****************************************************//*
 * @file        main.c
 * @brief       Contains function for the audio codec project aic32x4
 * @version     1.0
 * @date        09.02.2012
 * @author      Heiko Polster
 ************************************************************************/
#include "audio.h"
#include "delay.h"
#include "lcd.h"

unsigned int iIn;

static double data0[3];      // data field for basic wave
static double data1[3];      // data field for 1st high wave
static double data2[3];      // data field for 2nd high wave

// ---------------------------------------------------------------------
// init for resonator start
// ---------------------------------------------------------------------
void init_square(void)
{
    data0[0] = data1[0] = data2[0] = -1;
    data0[1] = data1[1] = data2[1] = 0;
    data0[2] = data1[2] = data2[2] = 1;
}

// ---------------------------------------------------------------------
// function with 3 digital resonators with 100Hz, 300Hz, 500Hz
// ---------------------------------------------------------------------
short int square(void)
{
    short int out;

    static short int n=0,N;

    const short int A = 0x800;
    const float pi = 3.1415926;

    N = 8000/100;            // number of steps for one basic wave

    // coeffizients for sine waves: kx = 2 * cos(2 * PI * f0 / fs)
    const float k0 = 1.9938346; //  100Hz
    const float k1 = 1.9447398; //  300Hz
    const float k2 = 1.8477590; //  500HZ

    // basic wave = 100Hz
    data0[0] = (data0[1] * k0 - data0[2]);

    data0[2] = data0[1];

    data0[1] = data0[0];


    // 1st high wave = 3 * basic wave
    data1[0] = (data1[1] * k1 - data1[2]);

    data1[2] = data1[1];

    data1[1] = data1[0];

    // 2nd high wave = 5 * basic wave
    data2[0] = (data2[1] * k2 - data2[2]);
```

```
    data2[2] = data2[1];

    data2[1] = data2[0];

                        // sinus  + 1st       + 2nd high wave
    out = (short int) (4/pi*(data0[0] + data1[0] + data2[0]))* A;   // sine

    n++;                // step counter + 1

    if(n>(N-1))         // if one basic wave
    {
        n=0;            // step counter = 0

        init_square();  // set start values for field elements
    }

    return out;         // return with new value
}

void square_filter(int16_t* l_ch, int16_t* r_ch) {
    *l_ch = square();
    *r_ch = 0;
}


/*****************************************************************************//**
 * @brief       main function
 * @param[in]   none
 * @return      none
 *****************************************************************************/
int main(void)
{
    LCD_INIT();                                 // LCD init
    LCD_gotoXY(1,0);                            // column 1, char 0
    LCD_SENDSTRING("square");                       // send text to LCD
    delay_ms(1);                                // wait 1 ms

    init_square();                              // init the sine generator

    audio_I2C_init();                           // init I2C interface for codec
    audio_init();                               // init audio interface

    audio_set_filter(square_filter);                // set sine function for signal

    while(1)                                    // endless loop
    {
        delay_ms(200);                          // wait 200 ms
    }
}
```