Keypad Matrix:

```
#include <pic18f4550.h>
#define LCD_EN LATAbits.LA1
#define LCD_RS LATAbits.LA0
#define LCDPORT LATB

const unsigned char KeyLookupTbl[]={'1','4','7','.',
       '2','5','8','0',
   '3','6','9','#',
   'A','B','C','D'};

void lcd_delay(unsigned int time)
{
 unsigned int i , j ;

    for(i = 0; i < time; i++)
    {
            for(j=0;j<50;j++);
    }
}


void SendInstruction(unsigned char command)
{
     LCD_RS = 0; // RS low : Instruction
     LCDPORT = command;
     LCD_EN = 1; // EN High
     lcd_delay(10);
     LCD_EN = 0; // EN Low; command sampled at EN falling edge
     lcd_delay(10);
}

void SendData(unsigned char lcddata)
{
     LCD_RS = 1; // RS HIGH : DATA
     LCDPORT = lcddata;
     LCD_EN = 1; // EN High
     lcd_delay(10);
     LCD_EN = 0; // EN Low; data sampled at EN falling edge
     lcd_delay(10);
}

void InitLCD(void)
{
    ADCON1 = 0x0F;
    TRISB = 0x00; //set data port as output
    TRISAbits.RA0 = 0; //RS pin
    TRISAbits.RA1 = 0; // EN pin

    SendInstruction(0x38);      //8 bit mode, 2 line,5x7 dots
    SendInstruction(0x06); // entry mode
    SendInstruction(0x0C); //Display ON cursor OFF
    SendInstruction(0x01);      //Clear display
    SendInstruction(0x80);      //set address to 0
}



/*Reads a Single key*/
//Returns the ascii value of the key
unsigned char ReadKey(void)
{
 unsigned char row,val, i, j, key=0;

 while(1)                                   //Loop till a key is
pressed
 {
   LATD = 0xFF;
   for(i=0x01;i<0x10;i=i<<1)
   {
       LATD = ~i;                           //Make output pin of PORTD
low, one column at a time
       lcd_delay(2);
       row = PORTD>>4;                      //Scan rows
       for(j=0x01;j<0x10;j=j<<1)
       {
           if((row & j) == 0)                 //Check which row
scanned is low
           {
               val = KeyLookupTbl[key];     //If a key is pressed,
find and return the corresponding character
               return val;
           }
           else
               key++;                       //If key not pressed,
increament key counter value
       }
   }

 }
}

void main()
{
    unsigned char Key, str[16];
    unsigned char *string1 = "Key Pressed = ";

    TRISD = 0xF0;                           //rows as inputs and
columns as output
    LATD = 0xFF;

    InitLCD();
    SendInstruction(0x80);                  //set 1st line
    while(*string1)
        SendData(*string1++);


     while(1)                               //Forever loop
    {
        Key = ReadKey();                    //Check the key pressed
        SendInstruction(0xC0);              //set 2nd line
        SendData(Key);
        lcd_delay(100);
     }
}
```

7 Segment Display:

0 to 9:

```
#include <pic18f4550.h>

// Segment bit definitions for common cathode 7-segment display
#define ONE     0b0000110
#define TWO     0b1011011
#define THREE   0b1001111
#define FOUR    0b1100110
#define FIVE    0b1101101
#define SIX     0b1111101
#define SEVEN   0b0000111
#define EIGHT   0b1111111
#define NINE    0b1101111
#define ZERO    0b0111111

const unsigned char segData[10] = {
    ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE
};

// Simple delay function
void delay()
{
    unsigned int i,j;
    for(i = 0; i < 500; i++){
        for(j = 0; j < 500; j++);
}
}

int main(void)
{
    unsigned char count;

    TRISB = 0x00;   // Set PORTB as output for segments
    LATB = 0x00;    // Clear PORTB output

    while (1)
    {
        for (count = 0; count < 10; count++)  // Loop from 0 to 9
        {
            LATB = ~segData[count];  // Send pattern to 7-seg (invert
if needed)
            delay();              // Wait before next number
        }
    }

    return 0;
}
```

0 to 100:

```
#include <pic18f4550.h>

#define ONE     0b0000110
#define TWO     0b1011011
#define THREE   0b1001111
#define FOUR    0b1100110
#define FIVE    0b1101101
#define SIX     0b1111101
#define SEVEN   0b0000111
#define EIGHT   0b1111111
#define NINE    0b1101111
#define ZERO    0b0111111

#define SEG1    LATAbits.LATA0
#define SEG2    LATAbits.LATA1

const unsigned char segData[10] = {
    ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE
};

void delay(unsigned int time)
{
    int i;
    while (time--)
        for (i = 0; i < 500; i++);
}

int main(void)
{
    unsigned char count = 0;
    unsigned char digit1, digit2;
    unsigned int i;

    TRISB = 0x00;
    LATA = 0x00;
    TRISAbits.TRISA0 = 0;
    TRISAbits.TRISA1 = 0;

    while (1)
    {
        for (count = 0; count < 100; count++)
        {
            digit1 = count / 10;
            digit2 = count % 10;

            for (i = 0; i < 100; i++)
            {
                SEG1 = 1;
                SEG2 = 0;
                LATB = ~segData[digit1];
                delay(2);

                SEG1 = 0;
                SEG2 = 1;
                LATB = ~segData[digit2];
                delay(2);
            }
        }
    }

    return 0;
}
```

Relay Buzzer:

```
#include <PIC18F4550.h>
void delay()
{
    unsigned int i;
    for(i = 0; i < 30000; i++);
}
void main()
{
    unsigned char i, key = 0;
    TRISB = 0x00;      // PORTB as output (for LEDs)
    LATB = 0x00;       // Clear PORTB
    ADCON1 = 0x0F;     // Set PORTA pins as digital
    TRISAbits.TRISA2 = 1;  // Set RA2 as input (Button 1)
    TRISAbits.TRISA3 = 1;  // Set RA3 as input (Button 2)
    TRISAbits.TRISA5 = 0;  // Set RA5 as output (Buzzer/Relay)
    TRISAbits.TRISA4 = 0;  // Set RA4 as output (Relay/Buzzer)
    while(1)
    {
        // If button pressed on RA2  key = 0
        if(PORTAbits.RA2 == 0) key = 0;
        // If button pressed on RA3  key = 1
        if(PORTAbits.RA3 == 0) key = 1;
        if(key == 0)
        {
            LATAbits.LATA4 = 1;   // Turn ON RA4
            LATAbits.LATA5 = 0;   // Turn OFF R
            // Left to right LED chase
            for(i = 0; i < 8; i++)
            {
                LATB = 1 << i;
                delay();
                LATB = 0x00;
                delay();
            }
        }
        if(key == 1)
        {
            LATAbits.LATA4 = 0;   // Turn OFF RA4
            LATAbits.LATA5 = 1;   // Turn ON RA5
            // Right to left LED chase
            for(i = 7; i > 0; i--)
            {
                LATB = 1 << i;
                delay();
                LATB = 0x00;
                delay();
            }
        }
    }
}
```

Square Wave:

```
#include <pic18f4550.h>

volatile unsigned char timer_set = 0;
void timerInit(void) {
// Timer0 configuration: 16-bit mode, prescaler 1:256
T0CON = 0b00000111; // 16-bit, prescaler 1:256, timer off initially
TMR0H = 0xED; // Load timer high byte
TMR0L = 0xB0; // Load timer low byte
}
void Interrupt_Init(void) {
RCONbits.IPEN = 1; // Enable interrupt priority
INTCONbits.GIE = 1; // Enable global interrupts
INTCONbits.PEIE = 1; // Enable peripheral interrupts (optional)
INTCONbits.TMR0IE = 1; // Enable Timer0 interrupt
INTCONbits.TMR0IF = 0; // Clear Timer0 interrupt flag
INTCON2bits.TMR0IP = 0; // Set Timer0 interrupt as low priority
}
void __interrupt(low_priority) LowISR(void) {
if (INTCONbits.TMR0IF) {
T0CONbits.TMR0ON = 0;
INTCONbits.TMR0IF = 0;
TMR0H = 0xED;
TMR0L = 0xB0;
LATB = ~LATB;
T0CONbits.TMR0ON = 1;
}
}
void main(void) {
TRISB = 0x00; // Set PORTB as output
LATB = 0xFF; // Initialize PORTB pins to high
Interrupt_Init(); // Initialize interrupts
timerInit(); // Initialize timer
T0CONbits.TMR0ON = 1; // Start Timer0
while(1) {
// Main loop does nothing, waiting for interrupts
}
}
```

Servo Motor:

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(2);  // attaches the servo on pin 9 to the servo
object
}

void loop() {
  for (pos = 0; pos <= 90; pos += 1) { // goes from 0 degrees to 180
degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position
in variable 'pos'
    delay(15);                       // waits 15ms for the servo to
reach the position
  }
  for (pos = 90; pos >= 0; pos -= 1) { // goes from 180 degrees to 0
degrees
    myservo.write(pos);              // tell servo to go to position
in variable 'pos'
    delay(15);                       // waits 15ms for the servo to
reach the position
  }
}
```