```c
/*         __                           __
          /'{>                         /'{>
      ____) (____                  ____) (____
    //'--;   ;--'\\             //'--;   ;--'\\
   ///////\_/\\\\\\\           ///////\_/\\\\\\\
          m m                          m m
        *********** PIC18F46K80 ***********
                +--------U--------+
       MCLR/RE3 |1              40| RB7  PICKit & LED Bit 7
      POT - RA0 |2              39| RB6  PICKit & LED Bit 6
Light Sens  RA1 |3              38| RB5  LED Bit 5
Analog 2    RA2 |4              37| RB4  LED Bit 4
Analog 3    RA3 |5              36| RB3  LED Bit 3
       VDD Core |6              35| RB2  LED Bit 2
Switch S2   RA5 |7              34| RB1  LED Bit 1
<not used>  RE0 |8              33| RB0  LED Bit 0 & Switch S5
<not used>  RE1 |9              32| VDD
<not used>  RE2 |10             31| GND
            VDD |11             30| RD7 (Rx2)-> Serial Data OUT*  Bluetooth
            GND |12             29| RD6 (Tx2)-> Serial Clock OUT* Bluetooth
  Switch S4 RA7 |13             28| RD5 To LCD
  Switch S3 RA6 |14             27| RD4 To LCD
  To LCD    RC0 |15             26| RC7 (Rx1)-> Serial Data OUT*  USB
  To LCD    RC1 |16             25| RC6 (Tx1)-> Serial Clock OUT* USB
  Buzzer    RC2 |17             24| RC5 <Not Used>
  I2C SCLK  RC3 |18             23| RC4 I2C SDA
  To LCD    RD0 |19             22| RD3 To LCD
  To LCD    RD1 |20             21| RD2 To LCD
                +-----------------+
* Note:  RA4 not available since used by VDD Core!
* for configurations of master synch serial port, see 18.3 of PDF
* YES - Rx IS SERIAL OUT!!!!
*/

#include <stdio.h>
#include <stdlib.h>
#include <p18f46K80.h>
#include <delays.h>
#include <timers.h>
#include <usart.h>
#include <i2c.h>

// see hlpPIC18ConfigSet document in the C18 director for info on configs below
#pragma config FOSC = INTIO2    /* internal osc enable */
#pragma config FCMEN = OFF      /* Failsafe Clock Monitor Disabled */
#pragma config WDTEN = OFF      /* no Dog */
#pragma config SOSCSEL = DIG    /* Configures PORTC0&1 for digital I/O */
#pragma config XINST = OFF      /* Extended Instruction Set off */

extern void LCDInit(void);      /* LCD initialize    */
extern void lcd_clr(void);      /* clear the lcd     */
extern void LCDLine_1(void);    /* get LCD to line 1 */
extern void LCDLine_2(void);    /* get LCD to line 2 */
extern void end_line(void);     /* end of line       */
extern void d_write(void);      /* write data to LCD */
extern void i_write(void);      /* position lcd curs.*/
extern unsigned int temp_wr;    /* pass ascii char   */

//****************************************************************
//*                                                              *
//*  ECNS-424                                                    *
//*  18 April 2021                                               *
//*  In Depth Lab-4                                              *
//*                                                              *
//*  The purpose of this program is to use Asynchronous          *
//*  communication via the on chip USART to an external serial   *
//*  monitor running on a lap or desktop computer                *
//*                                                              *
//*  Created by: Brandon Empie                                   *
//*                                                              *
//****************************************************************
//24 notes for dividing the SuperMario theme into two parts
int NoteArray[] =
{660,660,660,510,660,770,380,510,380,320,440,480,450,430,380,660,760,860,700,760,660,520,580,480};
int NoteLength[] = {20,20,20,20,20,20,20,20,20,20,20,16,20,20,20,16,10,20,16,10,16,16,16,16};
int delay[] = {30,60,60,20,60,110,115,90,80,100,60,66,30,60,40,40,30,60,30,70,60,30,30,100};
//21 notes for the second portion of SuperMario theme
const rom int NoteArrayA[] =
{500,760,720,680,620,650,380,430,500,430,500,570,500,760,720,680,620,650,1020,1020,1020};
const rom int NoteLengthA[] = {20,20,20,20,30,30,20,20,20,20,20,20,20,20,20,20,30,40,16,16,16};
const rom int delayA[] = {60,20,30,30,60,60,30,30,60,30,20,44,60,20,30,30,60,60,60,30,60};

void ADConverter(void);    //function prototype for subroutine "ADConverter"
void BlinkM(void);         //function prototype for subroutine "BlinkM"
void TempRead(void);       //function prototype for subroutine "TempRead"
void Color(void);          //function prototype for subroutine "Color"
void LEDs(void);           //function prototype for subroutine "LEDs"
void SendString(void);     //function prototype for subroutine "SendString"
void SuperMario(void);     //function prototype for subroutine "SuperMario"
void Compare(void);        //Compare ISR function prototype

unsigned char result;      //main switch statement variable
unsigned char BlinkMColor; //Blink-M switch statement variable
unsigned int Intensity;    //Blink-M intensity variable
unsigned long X = 0; //init. variable X, A/D result variable
unsigned long Y = 0; //init. variable Y, Voltage result variable
unsigned long Z = 0; //init. variable Z, Lux result variable
unsigned int R = 0; //initialize variable R
unsigned int G = 0; //initialize variable G
unsigned int B = 0; //initialize variable B
unsigned int H = 0; //Hundreds place BlinkM
unsigned int T = 0; //Tens place BlinkM
unsigned int O = 0; //Ones place BlinkM
unsigned int HexUpper;//variable for user entered upper byte
unsigned int HexLower;//variable for user entered lower byte
char LCDString[16];    //Array for user entered Strings
unsigned int S = 0;    //String pointer variable
char done = 0;        //variable used to check if user has pressed Enter key
unsigned int ArraySize = 0; //keeps track of how many chars need to be sent to LCD
unsigned int j;     //Generic loop variable
unsigned int ready = 1;//Used to check if the TC-74 Temperature Sensor was ready to be read
signed char temperature; //TC-74 temp. sensor read in variable
signed char read_data = 0; // Signed 8 bits -127 +128 deg Celcius.
unsigned char cmd = 1;  // Select TC74 Config Reg.

#pragma code high_vector=0x08   // tells compiler that this belongs @ loc 0x08
void high_interrupt (void)
{
  _asm GOTO Compare _endasm // inline assembly for interrupt vector
}
#pragma code              // normal code area from here down
#pragma interrupt Compare //declare interrupt Compare code to follow

//****************************************************************
//*                                                              *
//*  ISR: Compare                                                *
//*  In:  none                                                   *
//*  Out: none                                                   *
//*                                                              *
//*  The purpose of this Interrupt Service Routine is to create  *
//*  the generated frequency via capture compare                 *
//*                                                              *
//*  Created by: Brandon Empie                                   *
//*                                                              *
//****************************************************************
void Compare(void)
{
    if(PIR3bits.CCP2IF = 1) // check to see if CCP2IF caused the interrupt
    {
    TMR1H = 0x00;           //Clear Timer 1 high byte
    TMR1L = 0x00;           //Clear Timer 1 Low byte
    PIR3bits.CCP2IF = 0;    //Clear CCP2IF Flag for next go-round
    }
}
//////////////////////////////////////////////////////////////////
//                          MAIN                                //
//////////////////////////////////////////////////////////////////
void main (void)
{
    //init I/O below
    OSCCON = 0x60;  //set to 8 MHz base clock internal (default clock)
    TRISB = 0;      //PORT B all outputs
    TRISC = 0xF9;   //PORT C config
    TRISA = 0xFF;   //PORTA all inputs
    LATB = 0;       //All outputs off for safety

    PORTAbits.RA1 = 1;    //Light sensor channel set to analog
    ADCON0bits.CHS = 1;   //A/D now set to channel 01 (AN1) for light sensor
    ADCON2bits.ADFM = 0;  //Left Justify A/D
    ADCON2bits.ADCS = 4;  //FOSC/4
    ADCON2bits.ACQT = 2;  //A/D acquisition time select 4 TAD
    ADCON0bits.ADON = 1;  //Turn on A/D

    OpenI2C(MASTER, SLEW_OFF); //Init I2C
    TRISCbits.TRISC3 = 1;   //I2 C clock output (MSSP module); takes priority over port data.
    TRISCbits.TRISC4 = 1;   //I2C data input/output (MSSP module); input type depends on module setting.
```

```c
    SSPCON1 = 0x28;         //SSP enable(bit5)...Master mode(bit3-0)->0000 = SPI Master mode, clock = FOSC/4
    SSPSTATbits.SMP = 1;    //Input data sampled at end of data output time
    SSPSTATbits.CKE = 1;    //Transmit occurs on transition from active to Idle clock state
    SSPADD = 0x27;          //sets up baud rate at 50kHz = 8MHz/(4*(39 + 1))

    Open1USART (USART_TX_INT_OFF &
                USART_RX_INT_OFF &
                USART_ASYNCH_MODE &
                USART_EIGHT_BIT &        //Error rate = (9615-9600)/9600 = .16%
                USART_CONT_RX &          //Baud = 8MHz/(16(51+1))
                USART_BRGH_HIGH, 51);    // sets baud to 9600 @ 8 MHz
    BlinkM(); //call "blinkM" to turn blinkM off on startup

    TRISCbits.TRISC2 = 0;   //RC2 output for buzzer on CCP2
    PORTCbits.RC2 = 1;      //Buzzer channel set to analog
    CCP2CON = 0x02;         //toggle mode
    PIE3bits.CCP2IE = 1;    //enable compare interrupt
    RCONbits.IPEN = 1;      //enable priority interrupts (both low and hi)
    INTCONbits.GIEH = 1;    //enables high-priority interrupts
    IPR3bits.CCP2IP = 1;    //set Compare 2 interrupt to high priority
    T1CONbits.T1CKPS = 2;   //prescaler of 4 * Fosc/4 to get pre of 16
    T1CONbits.RD16 = 1;     //Timer 1 16-bit mode
    T1CONbits.TMR1ON = 0;   //Timer 1 off
    CCPR2 = ((250000/NoteLength[j])-3); //Init. CCPR2 = 250000/frequency - offset

    //Display startup prompt to the USART
    putrs1USART ((const far rom char *)"\n\rECNS-424 In Depth Lab #4, by Brandon Empie\n\r");
    putrs1USART ((const far rom char *)"\rWaiting for a command...\n\r\r");

    LCDInit();      //Initialize LCD
    lcd_clr();      //clear the display

    while (1)
    {
        while(!DataRdy1USART()); //poll till USART ready
        result = getc1USART();   //Result = Read USART char byte
        switch(result)
        {
            case 'L' :                //Return Lux value if L is entered
                ADConverter();        //call "ADConverter"
                Y = ((X*500)/255);
                //Y(Vin) = ((A/D result)* 500)/255(8-bit)
                //(5V, 500 causes Y value to be ex. 320 instead of 3.2 to bypass floating point)
                Z = ((Y*1000)/224);   // Z now = Lux (lux value is based on 2.240mV/Lux)
                fprintf(_H_USART, "Current Lux value is:  %lu\n\r\r", Z);//Display lux value to the USART
                break;//break out of switch statement
            case 'B' :                //accept color and R,G,B intensity (0-255)
                Color();              //Call "Color"
                break;                //break out of switch statement
            case 'D' :                //User selects hex byte to be displayed on PIC LEDs
                LEDs();               //Call "LEDs"
                break;                //break out of switch statement
            case 'N' :
                //User given option of static note or SuperMario theme song, next line displayed to USART
                putrs1USART ((const far rom char *)
                "\n\rPress 'N' again to turn on note or 'S' to play SuperMario theme song\n\r\r");
                while(!DataRdy1USART()); //poll till USART ready
                j = getc1USART();        //j = Read USART char byte
                switch(j)
                {
                    case 'N' : //Turn on static note
                        CCPR2 = ((250000/659.256)-3); //CCPR2 = 250000/frequency - offset Note E Natural
                        T1CONbits.TMR1ON = 1;   //Timer 1 on
                        //Display prompt to the USART
                        putrs1USART ((const far rom char *)"\n\rNote E natural on\n\r");
                        break;//break out of switch statement
                    case 'S' :  //play SuperMario
                        SuperMario();   //Call "SuperMario"
                        break;//break out of switch statement
                    default://in case user does not follow directions, Display next prompt to the USART
                        putrs1USART ((const far rom char *)"\n\rInvalid Response Try Again\n\r");
                }
                break;//break out of switch statement
            case 'O' :  //Turn off static note
                T1CONbits.TMR1ON = 0;   //Timer 1 off
                putrs1USART ((const far rom char *)"\n\rNote off\n\r"); //Display prompt to the USART
                break;//break out of switch statement
            case 'S' : //Display next prompt to the USART, user selects string for LCD
                putrs1USART ((const far rom char *)"\n\rEnter String for Line 1, Press Enter when done\n\r\r");
                lcd_clr();      //clear the display
                LCDLine_1();    //Setup 1st line
                SendString();   //call "SendString"
                putrs1USART ((const far rom char *)
                "\n\rEnter a String for Line 2? Press 'y' for yes, or 'n' for No\n\r\r");
                while(!DataRdy1USART()); //poll till USART ready
                j = getc1USART();   //j = Read USART char byte
                if(j == 'y' || j == 'Y') //if user enters a lower case or upper case Y
                {
                   putrs1USART ((const far rom char *)
                   "\n\rEnter String for Line 2, Press Enter when done\n\r\r");
                   LCDLine_2();//Set cursor at the start of line 2
                   SendString();//call "SendString"
                }
                break;//break out of switch statement
            case 'T' ://Read temperature in Centigrade and display to USART
                TempRead(); //call "TempRead"
                fprintf(_H_USART, "Current Temperature is %hhi Degrees Centigrade\n\r\r", temperature);
                break;//break out of switch statement
            default ://if user enters invalid command, display commands available
                putrs1USART ((const far rom char *)"\n\rEnter L to return current Lux value\n\r");
                putrs1USART ((const far rom char *)"\n\rEnter B to set Blink-M color and intensity\n\r");
                putrs1USART ((const far rom char *)"\n\rEnter D to set byte value of LEDs\n\r");
                putrs1USART ((const far rom char *)"\n\rEnter N to turn on musical Note\n\r");
                putrs1USART ((const far rom char *)"\n\rEnter O to turn off musical note\n\r");
                putrs1USART ((const far rom char *)"\n\rEnter S to send a string to BullDogPIC++ LCD\n\r");
                putrs1USART ((const far rom char *)
                "\n\rEnter T to return current temperature in degrees Centigrade\n\r");
        }
    //display prompt when ready
    putrs1USART ((const far rom char *)"\n\rWaiting for a command...\n\r\r");
    }
}

//****************************************
//* Subroutine: ADConverter              *
//*                                      *
//* Inputs: none                         *
//* Outputs: none                        *
//*                                      *
//* The purpose of this subroutine is to *
//* read the A/D placing the result in   *
//* variable X                           *
//*                                      *
//* Created by: Brandon Empie            *
//****************************************
void ADConverter(void)
{
   ADCON0bits.GO = 1;          //Start A/D
   while(ADCON0bits.GO);       //wait till A/D is done

   X = ADRESH;                 //X = A/D result 8-bit

   return;                     //return to main
}
//****************************************
//* Subroutine: BlinkM                   *
//*                                      *
//* Inputs: none                         *
//* Outputs: none                        *
//*                                      *
//* The purpose of this subroutine is to *
//* set the color and intensity of the   *
//* Blink M using i2c                    *
//*                                      *
//* Created by: Brandon Empie            *
//****************************************
void BlinkM(void)
{
    IdleI2C();//idle i2c
    StartI2C();//Start i2c
    while(SSPCON2bits.SEN); //wait for start

    while(WriteI2C(0x00));  //BlinkM address
    IdleI2C();//idle i2c

    while(WriteI2C('o'));  // send BlinkM command to stop script
    IdleI2C();//idle i2c

    while(WriteI2C('n'));  // send BlinkM command to send RGB values below
    IdleI2C();//idle i2c

    while(WriteI2C(R));  // send BlinkM value for Red
    IdleI2C();//idle i2c

    while(WriteI2C(G));  // send BlinkM value for Green
    IdleI2C();//idle i2c

    while(WriteI2C(B));  // send BlinkM value for Blue
    IdleI2C();//idle i2c

    SSPCON2bits.PEN = 1;    // initiate STOP
    while(SSPCON2bits.PEN); // Wait for STOP mode idle

    StopI2C();//stop i2c
    while(SSPCON2bits.PEN);//wait for stop condition to complete

    return;
}
//*****************************************
//* Subroutine: TempRead                  *
//*                                       *
//* Inputs: none                          *
//* Outputs: none                         *
//*                                       *
//* The purpose of this subroutine is to  *
//* read the TC-74 temp sensor and save   *
//* the value in temperature variable     *
//*                                       *
//* Created by: Brandon Empie             *
//*****************************************
void TempRead(void)
{
while(ready == 1) //run loop until temp has been read
    {
    StartI2C();             //Start I2C
    while(SSPCON2bits.SEN); // wait for start

    while(WriteI2C(0x9A));  // Send TC74 address (Write; LSB=0)
                                // function returns a zero if success, i.e ack.
                                // see C18 Lib PDF pp. 28
// the command is placed in the variable cmd below.
// 1 = read config, 0 = read temperature.

    while(WriteI2C(cmd));   // Select TC74 Reg 1 (Config)
                                // wait for ack from slave
    IdleI2C();              // idle i2c
    RestartI2C();           // Restart i2c
    IdleI2C();              // idle i2c

    while(WriteI2C(0x9B));  // Send TC74 address (Read; LSB=1)
                                // wait for ack from slave

    read_data = ReadI2C();  // Read register from TC74

    NotAckI2C();            //not acknowlege

    IdleI2C();              // idle i2c

    StopI2C();              // stop i2c
// at this point, the variable read_data has the response from the TC74

// the next step would be to determine whether the TC74 is ready to
// read temperature, then go through the sequence again based on that
// response to either read the temp, or to read the config again?

    if(cmd == 0)            // SET a breakpoint here & watch read_data
    {
        temperature = read_data; //copy read_data into temperature variable
        ready = 0;       //once temperature has been read set ready to 0, to end subroutine loop
    }
    cmd = read_data & 0x40?0:1; //if ready for reading cmd updated with 0
    }
    ready = 1; //Set ready to 1 for next go round

    return;
}
//*****************************************
//* Subroutine: Color                     *
//*                                       *
//* Inputs: none                          *
//* Outputs: none                         *
//*                                       *
//* The purpose of this subroutine is to  *
//* set the Blink-M color based on user   *
//* input                                 *
//*                                       *
//* Created by: Brandon Empie             *
//*****************************************
void Color(void)
```

```
{
//Display prompts to USART
putrs1USART ((const far rom char *)"\rEnter 'R' for Red, 'G' for Green, or 'B' for Blue\n\r");
putrs1USART ((const far rom char *)"\rOr 'Y' for Yellow, 'C' for Cyan, 'P' for Purple\n\r\r");
while(!DataRdy1USART());    //poll till USART ready
BlinkMColor = getc1USART(); //BlinkMColor = get char using 1USART
putrs1USART ((const far rom char *)"\n\rEnter Blink-M Intensity from (0-255)\n\r");
putrs1USART ((const far rom char *)"\n\rFor example Intensity of '5' Enter '005'\n\r\r");
while(!DataRdy1USART());//poll till USART ready
H = getc1USART() - 48; //parse USART value into hundreds place
while(!DataRdy1USART());//poll till USART ready
T = getc1USART() - 48;  //parse USART value into Tens place
while(!DataRdy1USART());//poll till USART ready
O = getc1USART() - 48;  //parse USART value into ones place
Intensity = (H*100 + T*10 + O); //intensity = user entered three digit number
if(Intensity > 255) //If user enters in an intensity over 255
    BlinkMColor = X;//Set BlinkMColor to X to force default
switch(BlinkMColor)//outcome based on user input
{
    case 'R' ://if user selects Red
        R = Intensity;//Set variables accordingly for user desired output
        G = 0x00;
        B = 0x00;
        BlinkM();//call "BlinkM"
        break;//break out of switch statement
    case 'G' ://if user selects Green
        R = 0x00;//Set variables accordingly for user desired output
        G = Intensity;
        B = 0x00;
        BlinkM();//call "BlinkM"
        break;//break out of switch statement
    case 'B' ://if user selects Blue
        R = 0x00;//Set variables accordingly for user desired output
        G = 0x00;
        B = Intensity;
        BlinkM();//call "BlinkM"
        break;//break out of switch statement
    case 'Y'://if user selects Yellow
        R = Intensity;//Set variables accordingly for user desired output
        G = Intensity;
        B = 0x00;
        BlinkM();//call "BlinkM"
        break;//break out of switch statement
    case 'C'://if user selects Cyan
        R = 0x00;//Set variables accordingly for user desired output
        G = Intensity;
        B = Intensity;
        BlinkM();//call "BlinkM"
        break;//break out of switch statement
    case 'P'://if user selects purple
        R = Intensity;//Set variables accordingly for user desired output
        G = 0x00;
        B = Intensity;
        BlinkM();//call "BlinkM"
        break;//break out of switch statement
    default:
        putrs1USART ((const far rom char *)"\n\rInvalid Response Try Again\n\r");
        R = 0x00;//turn off BlinkM if user does not follow directions
        G = 0x00;
        B = 0x00;
        BlinkM();//call "BlinkM"
}
return;                 //return to main
}
//*****************************************
//* Subroutine: LEDs                      *
//*                                       *
//* Inputs: none                          *
//* Outputs: none                         *
//*                                       *
//* The purpose of this subroutine is to  *
//* display a byte on the BulldogPIC++    *
//* LEDs as entered by the user           *
//*                                       *
//* Created by: Brandon Empie             *
//*****************************************
void LEDs(void)
{
    //Display prompts to USART
    putrs1USART ((const far rom char *)"\n\rEnter LED Hex Byte from (00-FF)\n\r");
    putrs1USART ((const far rom char *)"\rLetters must be capitalized, ex. 0A\n\r\r");
    while(!DataRdy1USART());//poll till USART ready
    HexUpper = getc1USART();//store upper byte
    if(HexUpper >= 0x41)    //if upper byte is a letter
        HexUpper = HexUpper - 55; //set HexUpper from ASCII to decimal (between 10-15)
    else
        HexUpper = HexUpper - 48; //else if HexUpper is a number
    HexUpper = HexUpper << 4;     //shift HexUpper to upper nibble

    while(!DataRdy1USART());//poll till USART ready
    HexLower = getc1USART();//store lower byte
    if(HexLower >= 0x41)    //if HexLower is a letter
        HexLower = HexLower - 55; //set HexLower from ASCII to decimal (between 10-15)
    else
        HexLower = HexLower - 48; //otherwise if HexLower is a number
    HexUpper = HexUpper + HexLower; //combine the two nibbles into HexUpper
    LATB = HexUpper; //Send HexUpper to LED
    fprintf(_H_USART, "Byte %X on LEDs\n\r\r", HexUpper);//send byte back to USART
   return;                     //return to main
}
//*****************************************
//* Subroutine: SendString                *
//*                                       *
//* Inputs: none                          *
//* Outputs: none                         *
//*                                       *
//* The purpose of this subroutine is to  *
//* allow the user to send strings to the *
//* BulldogPIC++ LCD                      *
//*                                       *
//* Created by: Brandon Empie             *
//*****************************************
void SendString(void)
{
    do
    {
    while(!DataRdy1USART());//poll till USART ready
    LCDString[S] = getc1USART();//USART char into string
    done = LCDString[S];//check entered char to see if user pressed enter key
    S = S + 1;//increment LCDString array variable
    }while(done != 0xd && S != 16);//continue reading and saving user entered string
    //until they press enter or string is 16 characters long (fills entire LCD row)

    ArraySize = S;//copy S into ArraySize
    if(done == 0xd)//if Enter has been pressed
        ArraySize = ArraySize - 1;//decrement ArraySize
    S = 0;//Clear LCDString array index variable for next time

    for (j=0;j<ArraySize;j++)
    {
    temp_wr = LCDString[j];// write one char. at a time
    d_write();      // Send to LCD
    }
    end_line();     // end of line call
    return;         //return to main
}
//*****************************************
//* Subroutine: SuperMario                *
//*                                       *
//* Inputs: none                          *
//* Outputs: none                         *
//*                                       *
//* The purpose of this subroutine is to  *
//* play the Super Mario Song             *
//* Created by: Brandon Empie             *
//*****************************************
void SuperMario(void)
{
    //TMR1H & TMR1L must be cleared to prevent note cutoff
    for (j=0;j<24;j++)
    {
        CCPR2 = ((250000/NoteArray[j])-3); //CCPR2 = 250000/frequency - offset
        TMR1H = 0x00;           //Clear Timer 1 high byte
        TMR1L = 0x00;           //Clear Timer 1 Low byte
        T1CONbits.TMR1ON = 1;   //Timer 1 on
        Delay10KTCYx(NoteLength[j]);//Delay based on NoteLength array
        T1CONbits.TMR1ON = 0;   //Timer 1 off
        Delay10KTCYx(delay[j]); //Delay based on delay array
    }
    for (j=8;j<24;j++)
    {
        CCPR2 = ((250000/NoteArray[j])-3); //CCPR2 = 250000/frequency - offset
        TMR1H = 0x00;           //Clear Timer 1 high byte
        TMR1L = 0x00;           //Clear Timer 1 Low byte
        T1CONbits.TMR1ON = 1;   //Timer 1 on
        Delay10KTCYx(NoteLength[j]);//Delay based on NoteLength array
        T1CONbits.TMR1ON = 0;   //Timer 1 off
        Delay10KTCYx(delay[j]); //Delay based on delay array
    }
    for (j=0;j<21;j++)
    {
        CCPR2 = ((250000/NoteArrayA[j])-3); //CCPR2 = 250000/frequency - offset
        TMR1H = 0x00;          //Clear Timer 1 high byte
        TMR1L = 0x00;           //Clear Timer 1 Low byte
        T1CONbits.TMR1ON = 1;   //Timer 1 on
        Delay10KTCYx(NoteLengthA[j]); //Delay based on NoteLengthA array
        T1CONbits.TMR1ON = 0;   //Timer 1 off
        Delay10KTCYx(delayA[j]);//Delay based on delayA array
    }
   return;                     //return to main
}
```