```
// this is code for spark fun chip ---  ECE 376 - Binary Clock -----------------
//  Name:
//  Date:
//  Description:
//
//------------------------------

// Global Variables

int BYTE2;
int ZERObyte;
int lastBYTE;

// Subroutine Declarations
#include <pic18.h>
#include "function.c"
#include "function.h"
//#include "lcd_portd.c"

// Bootloader Routine

// Main Routine







void main(void)
{
   unsigned long int i;


   unsigned int TIME;

   TRISA = 0;
   TRISB = 0xFF;    //THIS MAKES ALL PINS ON PORTB INPUT
   TRISC = 0x70;    //TRISC = 0xF0; // Make bits 0..2 of PORTC output and port 7, 0111 0000, 3-6 are input
   TRISD=0x00;      //THIS MAKES ALL PINS ON PORTD OUTPUT
   TRISD = 0;
   TRISE = 0;
   ADCON1 = 0x0F;
   PORTA = 0;
   PORTB = 0;
   PORTC = 0;
   PORTD = 0;
   PORTE = 0;
   TIME = 0;


// initial conditions
RC0=1;              // RESET is high
RC1=1;              // CLK is high
RC2=0;              // DATA is low
RC7=0               // setting rc7 as 0 because for some reason was going high by default

int STOP=0xFF;      //0xFF                      //Stops a play or record, etc
int ZERObyte=0x00;  // Northern Pintail         //FOR ADDRESS 0000 0000
int TESTB=0x80;     // Eastern Screech Owl      //FOR ADDRESS 0000 0001
int Third=0x40;     // Canadian Goose           //FOR ADDRESS 0000 0010
int FOURTH=0xC0;    // Bar Headed Goose         //FOR ADDRESS 0000 0011
int FIFTH=0x20;     // Bald EAgle               //FOR ADDRESS 0000 0100
int ADDone=0x00;    // American Kestrel         //FOR ADDRESS 0000 0101

int lastBYTE=0;
int RANDOM=0;
int PINTAIL=0;
```

```
   while(1) {
//You need to redefine your initial conditions
RC0=1;                  //RESET
RC1=1;                  //CLK
RC2=0;                  //RC2 is DATA

STOP=0xFF;          //0xFF      //Stops a play or record, etc
ZERObyte=0x00;
TESTB=0x80;     //FOR ADDRESS 0000 0001
Third=0x40;
FOURTH=0xC0;
FIFTH=0x20;
ADDone=0x00;
PINTAIL=0xA0;
/*
if(RB0=1)//random
{
RC5=0; Wait_ms(5);
RC5=1; Wait_ms(300);
RC6=0; Wait_ms(5);
//RC6=1; //clk is high again

//RANDOM = Random_Number();
//PORTD=RANDOM;

Wait_ms(500);
for(i=0; i<512; i++)
{
RC4=1;
Wait_ms(5);
if(RB0==1)
{
RANDOM=i;
}
RC4=0;
Wait_ms(5);
}



SPI_WriteByte(ZERObyte);        //sends 0x00
SPI_WriteByteSEVEN(RANDOM); //sends 0x00 - address 0 of 512

if(ZERO&1==1) {RC7=1; Wait_ms(1);} else{ RC7=0; Wait_ms(1);}
RC6=1;
Wait_ms(1);         //this wait command allows the clock to rise while data is still high,
                    //so that it isn't exactly on the rising edge

wait_noInterfere();
//makes others wait a certain number of ms without interferance

}
*/

if(RB1=1)//Northern Pintail
{
RC0=0; Wait_ms(5);      //toggle RESET high-low-high
RC0=1; Wait_ms(300);    //increasing post reset delay
RC1=0; Wait_ms(5);      //start clock with a 5 ms valley
//RC6=1; //clk is high again

SPI_WriteByte(ZERObyte);        //sends 0x00
SPI_WriteByteSEVEN(ZERObyte);   //sends 0x00 - address 0 of 512

if(ZERObyte&1==1) {RC2=1; Wait_ms(2);} else{ RC2=0; Wait_ms(1);}
RC1=1;
Wait_ms(1);         //this wait command allows the clock to rise while data is still high,
```

```
                    //so that it isn't exactly on the rising edge

RC2=0;
Wait_ms(3043);      // Specific Time related to this Sound.
//wait_noInterfere();
//makes others wait a certain number of ms without interferance

}


if(RB2=1)//Eastern Screech Owl
{
if (TESTB >>7 == 1) {lastBYTE =1;} else {lastBYTE=0;}
// if statement needed, SPI_WriteByteSEVEN doesn't return byte2

RC0=0; Wait_ms(5);      //toggle RESET high-low-high
RC0=1; Wait_ms(300);    //increasing post reset delay
RC1=0; Wait_ms(5);      //start clock with a 5 ms valley

SPI_WriteByte(ZERObyte);    //sends 0x00
SPI_WriteByteSEVEN(TESTB);  //sends ox80 - corresponds to address 0000 0001 - address 1

if(lastBYTE&1==1) {RC2=1; Wait_ms(1);} else{ RC2=0; Wait_ms(1);}
RC1=1;
Wait_ms(1);         // this wait command allows the clock to rise while data is still high,
                    // so that it isn't exactly on the rising edge
RC2=0;              // Data goes low again.
Wait_ms(3103);      // Specific Time related to this Sound.

}

if(RB3=1)//Canadian Goose
{
RC0=0; Wait_ms(5);      //toggle RESET high-low-high
RC0=1; Wait_ms(300);
RC1=0; Wait_ms(5);
//RC6=1; //clk is high again
//Third=0x40;

SPI_WriteByte(ZERObyte);        //sends 0x00
SPI_WriteByteSEVEN(Third);  //sends 0x40 - corresponds to 0000 0010 - address 3

if(Third&1==1) {RC2=1; Wait_ms(1);} else{ RC2=0; Wait_ms(1);}
RC1=1;
Wait_ms(1);         //this wait command allows the clock to rise while data is still high,
                    //so that it isn't exactly on the rising edge
RC2=0;
Wait_ms(3554);      // Specific Time related to this Sound.
//wait_noInterfere();
//makes others wait a certain number of ms without interferance
}

if(RB4=1)//Bar-Headed Goose
{
if (FOURTH >>7 == 1) {lastBYTE =1;} else {lastBYTE=0;} // if statement needed, SPI_WriteByteSEVEN doesn't return ↙
     byte2
//FOURTH=0xC0;
RC0=0; Wait_ms(5);      //toggle RESET high-low-high
RC0=1; Wait_ms(300);
RC1=0; Wait_ms(5);
//RC6=1; //clk is high again


SPI_WriteByte(ZERObyte);            //sends 0x00
SPI_WriteByteSEVEN(FOURTH);     //sends 0xC0 - corresponds to 0000 0011 - address 4
if(lastBYTE&1==1) {RC2=1; Wait_ms(1);} else{ RC2=0; Wait_ms(1);}
RC1=1;
Wait_ms(1);         //this wait command allows the clock to rise while data is still high,
                    //so that it isn't exactly on the rising edge
```

```
RC2=0;
Wait_ms(2487);      // Specific Time related to this Sound.
//wait_noInterfere();
//makes others wait a certain number of ms without interferance
}

if(RB5=1)//Bald Eagle
{
if (FIFTH >>7 == 1) {lastBYTE =1;} else {lastBYTE=0;} // if statement needed, SPI_WriteByteSEVEN doesn't return ↙
    byte2
//FIFTH=0x20;
RC0=0; Wait_ms(5);      //toggle RESET high-low-high
RC0=1; Wait_ms(300);
RC1=0; Wait_ms(5);
//RC6=1; //clk is high again

SPI_WriteByte(ZERObyte);            //sends 0x00
SPI_WriteByteSEVEN(FIFTH);      //sends 0x20 - corresponds to 0000 0100 - address 4
if(lastBYTE&1==1) {RC2=1; Wait_ms(1);} else{ RC2=0; Wait_ms(1);}
RC1=1;
Wait_ms(1);         //this wait command allows the clock to rise while data is still high,
                    //so that it isn't exactly on the rising edge
RC2=0;
Wait_ms(2716);      // Specific Time related to this Sound.
//wait_noInterfere();
//makes others wait a certain number of ms without interferance
}

if(RB6=1)// American Kestrel
{
if (PINTAIL >>7 == 1) {lastBYTE =1;} else {lastBYTE=0;} // if statement needed, SPI_WriteByteSEVEN doesn't      ↙
    return byte2
//FIFTH=0x20;
RC0=0; Wait_ms(5);      //toggle RESET high-low-high
RC0=1; Wait_ms(300);
RC1=0; Wait_ms(5);
//RC6=1; //clk is high again

SPI_WriteByte(ZERObyte);            //sends 0x00
SPI_WriteByteSEVEN(PINTAIL);        //sends 0x20 - corresponds to 0000 0100 - address 4
if(lastBYTE&1==1) {RC2=1; Wait_ms(1);} else{ RC2=0; Wait_ms(1);}
RC1=1;
Wait_ms(1);         //this wait command allows the clock to rise while data is still high,
                    //so that it isn't exactly on the rising edge
RC2=0;
Wait_ms(2419);      // Specific Time related to this Sound.
//wait_noInterfere();
//makes others wait a certain number of ms without interferance
}




}

}
//}
//*****************************************************/

void SendClk_High(void) {
RC0=0; Wait_ms(5);
RC0=1; Wait_ms(300);
RC1=0; Wait_ms(5);
//RC6=1; //clk is high again
}


void SPI_WriteByte(int byte2, int i) {
for(i=0; i<8; i++){
```

```
if(byte2&1==1) {RC2=1; Wait_ms(1);} else{ RC2=0; Wait_ms(1);}   // if byte2 AND 1 = 1, DATA is 1, if not DATA is ↙
    0
RC1=1;          // set CLK high
Wait_ms(1);     // wait 1 ms
RC1=0;          // set CLK low
Wait_ms(1);     // wait 1 ms, these four lines creates the toggle step diagram for CLK.
byte2=byte2>>1; // this takes byte2, a binary number, and moves a zero on the left end
                // for example 1111 0000, would become 0111 1000
                // this allows the variable to represent a binary waveform you want to send, allows the code to ↙
    send it
}
}

void SPI_WriteByteSEVEN(int byte2, int i) {
for(i=0; i<7; i++){
if(byte2&1==1) {RC2=1; Wait_ms(1);} else{ RC2=0; Wait_ms(1);}   // if byte2 AND 1 = 1, DATA is 1, if not DATA is ↙
    0
RC1=1;          // set CLK high
Wait_ms(1);     // wait 1 ms
RC1=0;          // set CLK low
Wait_ms(1);     // wait 1 ms, these four lines creates the toggle step diagram for CLK.
byte2=byte2>>1; // this takes byte2, a binary number, and moves a zero on the left end
                // for example 1111 0000, would become 0111 1000
                // this allows the variable to represent a binary waveform you want to send, allows the code to ↙
    send it
}
}

//void wait_noInterfere()
/*{
RC7=0;
Wait_ms(3000);*/
//so no sound can interfere for 5 seconds


/*
int Random_Number(void)
{
int randback;
Wait_ms(500);
if(rand > 0)
{
rand=rand+1;
if(RB0==1)
{
randback=rand;
rand=0;
}
}
randback = 0xFF;
return(randback);   //you need parenthesis around it, you must have it
}
*/
```