

```

//Bryan Marek
//Mini-Mozarts (D1)
//11/15/2006
//This program runs all of the control functionality for the
//Mini-Mozarts Xylophonic Keyboard

#include <system.h>
#pragma CLOCK_FREQ 20000000
#pragma DATA 0x2007, _HS_OSC & _WDT_OFF & _LVP_OFF

//Pin Settings on Microcontroller
//C0 - KEY_CON_0
//C1 - KEY_CON_1
//C2 - SLIDE_CON_0
//C3 - SLIDE_CON_1
//C4 - KEY/SOL_SELECT
//C5 - KEY_IN
//C6 - ERROR_OUT
//B5 - PLAY
//B6 - LISTEN
//B7 - DEMO
//A0 - SLIDER_1
//A1 - SLIDER_2
//A2 - SLIDER_3
//A4 - ERROR_IN

//this function takes in the value of the note to be played,
//activates the proper channel on the decoder, and activates
//the solenoid to play the note

void play_note(int i){

    //setting pins C0 & C1 the binary channel corresponding to
    //the current note is set

    switch (i){

        case 0:        //dead
        {
            //0=0000b
            clear_bit(portc,0); //clear pin C0
            clear_bit(portc,1); //clear pin C1
        }
    }
}

```

```

        break;

    case 1:        //key C
    {
        //1=0001b
        set_bit(portc,0);    //set pin C0
        clear_bit(portc,1);  //clear pin C1
    }
    break;

    case 2:        //key D
    {
        //2=0010b
        clear_bit(portc,0);  //clear pin C0
        set_bit(portc,1);    //set pin C1
    }
    break;

    case 3:        //key E
    {
        //3=0011b
        set_bit(portc,0);    //set pin C0
        set_bit(portc,1);    //set pin C1
    }

} //end switch
delay_ms(500);
clear_bit(portc,0);
clear_bit(portc,1);
delay_ms(500);

} //end play_note()

//this function takes in the value of the note that will be
//played by the user and waits for input on the respective
//channel while causing an error tone when the wrong key
//is pressed before advancing

void read_note(int i){

    int j = 1;
    //setting pins C0 & C1 the binary channel corresponding to
    //the current note is set

    switch (i) {

```

```

case 0:      //dead
{
    //0=0000b
    clear_bit(portc,0); //set pin C0
    clear_bit(portc,1); //set pin C1
}
break;

case 1:      //key C
{
    //1=0001b
    set_bit(portc,0); //set pin C0
    clear_bit(portc,1); //set pin C1
    while(j){ //create an infinite loop

        if(portc.5 == 0) //wait for low level on pin C5
            j = 0; //listen button was hit
        while(porta.4 == 0) //wrong button was hit
            set_bit(portc,6); //output error signal
        clear_bit(portc,6); //turn off error signal

    }
}
break;

case 2:      //key D
{
    //2=0010b
    clear_bit(portc,0); //set pin C0
    set_bit(portc,1); //set pin C1
    while(j){ //create an infinite loop

        if(portc.5 == 0) //wait for low level on pin C5
            j = 0; //listen button was hit
        while(porta.4 == 0) //wrong button was hit
            set_bit(portc,6); //output error signal
        clear_bit(portc,6); //turn off error signal

    }
}
break;

case 3:      //key E
{
    //3=0011b
    set_bit(portc,0); //set pin C0

```

```

        set_bit(portc,1);    //set pin C1
        while(j){          //create an infinite loop

            if(portc.5 == 0) //wait for low level on pin C5
                j = 0;      //listen button was hit
            while(porta.4 == 0) //wrong button was hit
                set_bit(portc,6); //output error signal
            clear_bit(portc,6); //turn off error signal
        }
    }

} //end switch

delay_ms(1000);
clear_bit(portc,0); //clear pin C0
clear_bit(portc,1); //clear pin C1
delay_ms(500);

} //end read_note()

//AD Conversion

short conversion(void)
{
    short ADresult;
    //combine contents of adresh and adresl into single short variable
    ADresult = adresh;
    ADresult = ADresult << 8 | adresl;
    //scale ADresult to 5000
    ADresult = 4*ADresult + (8*ADresult)/10 + (8*ADresult)/100 +
(8*ADresult)/1000;
    return ADresult;
}

short conv (int chan)
{
    short ADresult;
    //does A-D conversion on indicated channel and returns short
    //variable by running the conversion function
    volatile bit go_done@ADCON0.2;
    adcon1 = 10000010b;

    if (chan == 0)
    {
        adcon0 = 10000001b;
    }
}

```

```

    }
    if (chan == 1)
    {
        adcon0 = 10001001b;
    }
    if (chan == 2)
    {
        adcon0 = 10010001b;
    }

    delay_us(20);
    go_done = 1;
    while (go_done);
    ADresult = conversion ();
    return (ADresult);
}

```

*//this function takes in the value of the current slider
 //activates the proper channel on the multiplexor
 //and reads the current value from it, returning the note
 //to the main function*

```

short read_slider(int i){

    short ADresult;

    int note = 0;

    //Set channel for slider to be read
    //Pins C2 and C3 control the decoder selecting
    //The active slider

    switch (i){

        case 0:        //slider 0
        {
            //0=0000b
            clear_bit(portc,2);    //clear pin C2
            clear_bit(portc,3);    //clear pin C3
            ADresult = conv(0);
        }
        break;

        case 1:        //slider1

```

```

        {
            //1=0001b
            set_bit(portc,2);    //set pin C2
            clear_bit(portc,3);  //clear pin C3
            ADresult = conv(1);
        }
        break;

    case 2:        //slider 2
    {
        //2=0010b
        clear_bit(portc,2);    //clear pin C2
        set_bit(portc,3);      //set pin C3
        ADresult = conv(2);
    }

} //end switch

//Determine which location the slider is and
//Set the corresponding note
if(ADresult<100){
    note = 0;
}
else if(ADresult<2000){
    note = 1; //C
}
else if(ADresult<3500){
    note = 2; //D
}
else{
    note = 3; //E
}

return note;

} //end read_slider()

//this function occurs when the play button is pressed
//it increments through each slider, reads each value
//and plays each note that is read with a solenoid

void listen(){

    int i=0;                //keep track of current slider
    int note=0;             //keep track of current note

```

```

set_bit(portc,4); //Switch to solenoid use

while(i<3){          //go through all sliders

    note = read_slider(i); //read current slider
    play_note(note);      //play current note
    i++;
}

clear_bit(portc,2); //reset slider to 0
clear_bit(portc,3);
clear_bit(portc,4); //set to keyboard use

} //end listen()

//this function occurs when the play along button is pressed
//it increments through each slider, reads each value
//and waits for the user to hit each key before moving
//on to the next one

void play(){

    int i=0;          //keep track of current slider
    int note=0;      //keep track of current note

    while(i<3){     //go through all sliders
        note = read_slider(i); //read current slider
        read_note(note);      //wait for note to be played
        i++;
    } //end while()

    clear_bit(portc,2); //reset slider to 0
    clear_bit(portc,3);

} //end play()

//This function plays a demo song for the user
//without having to set sliders or play keys
//It plays the first few notes of Mary Had a Little
//Lamb - E-D-C-D-E-E-E

void demo(){

```

```

    set_bit(portc,4);    //turn on solenoid use
    play_note(3);
    play_note(2);
    play_note(1);
    play_note(2);
    play_note(3);
    play_note(3);
    play_note(3);
    clear_bit(portc,4); //turn on keyboard use
    read_note(3);
    read_note(2);
    read_note(1);
    read_note(2);
    read_note(3);
    read_note(3);
    read_note(3);

} //end demo()

//this function waits for the user to select either the play
//or play along buttons and then calls the proper functions
//to carry out the selected actions

void main(){

    trisa = 00011111b; //port A, to be used for analog use
    trisb = 11100000b; //port B, bits 5,6,7 input, rest outputs
    trisc = 00100000b;
    clear_bit(option_reg,7); //enables pull-up resistors on port B

    //initialize
    clear_bit(portc,0); //clear pin C0
    clear_bit(portc,1); //clear pin C1
    clear_bit(portc,2); //clear pin C2
    clear_bit(portc,3); //clear pin C3
    set_bit(portc,4); //set pin C4
    clear_bit(portc,6); //clear pin C6

    while(1){ //create an infinite loop for user mode selection

        while(portb.5 == 0) //wait for low level on pin B5
            play(); //Play button was hit

        while(portb.6 == 0) //wait for low level on pin B6

```

```
        listen();          //listen button was hit
    while(portb.7 ==0) //wait for low level on pin B7
        demo();          //Demo button was hit
    }
} //end main()
```