****

**Computer Engineering**

**Music Box**

**By: Hrutvik Patel**



**Table of Contents**

Initial Proposal1

Introduction/Project Description3

Theory of Operation4

Construction Details and Results6

Code8

Reflection10

Initial Proposal

**This a copy of our proposal which we wrote without much intuition on the music box project**

**Team Members : Hrutvik Patel, Monil Patel**

**Project Description/Criteria :**

For our interfacing project we’ve decided to build a computer controlled music box.

It will have 8 buttons which will play different sounds, 5 of which will represent notes on the treble clef and 3 of which will be ‘unique sounds’ with the help of diodes (as seen in the schematic below). To play the sound we will NOT be using an electric piezometer; instead we will be playing it on the computer speaker.

Upon doing some outside research on musical notes we’ve decided to use the notes on the treble clef, EGBDF for our sounds. (Depending on the time we have, we might try to add additional features such as LED’s lighting up once a key is pressed)

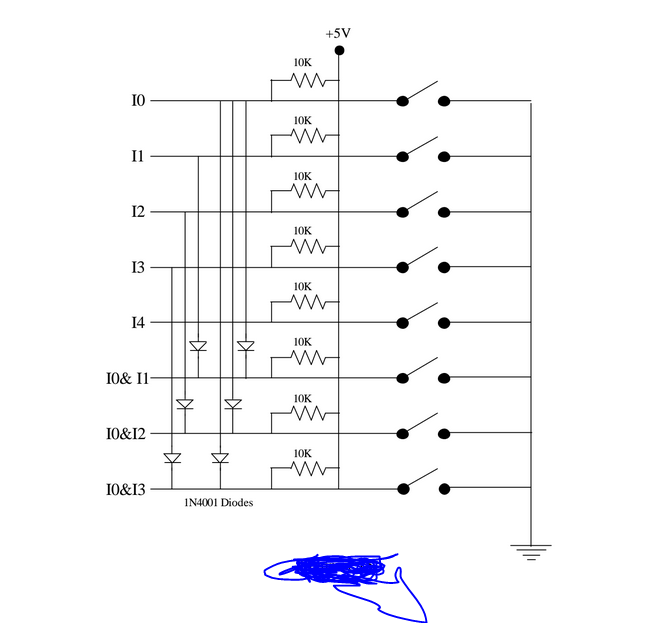
As for the programming portion of our project we are contemplating on which method to use. We haven’t experimented with sounds and frequencies using basic stamp in class, and the WAM pdf only shows how to make sound using a piezometer. As for using a parallel port and turing, only use a select few computers in our school can be used to interface (we know this from our previous house project).

The overall look of our project in terms of wiring will be like the schematic (using input pins and switches to trigger the sound). The look of our music box will be a rectangular prism like shape. And finally, our code will depend on what language we will use (basic stamp or turing). After confirming with Mr. Rosen that we are not allowed to use a piezometer we came to the conclusion of using turing for programming.

**Schematic:**

* There will also be a 74LS245 buffer chip that will  be used to prevent damage to the computer.
* Depending on the time we have we might add LEDs in series with the switches so they will light up when the switch is triggered.
* Interface cable would connect to the buffer chip and the computer.

Initial proposal



**Sketch:**

**refer to paper named ‘ISOMETRIC VIEW OF OUR MUSIC BOX’**

**Timeline:**

*MILESTONES*

1. *Creating a parallel port/interface cable and completion and testing of the electrical portion (¼, Friday May 20, 2015)*
2. *Completion and testing of the programming portion (½, Friday May 20, 2015)*
3. *Integrating the electrical portion to our music box and decorations (¾, Monday June 1, 2015)*
4. *Completion of report and final presentation (final, Thursday June 4, 2015)*

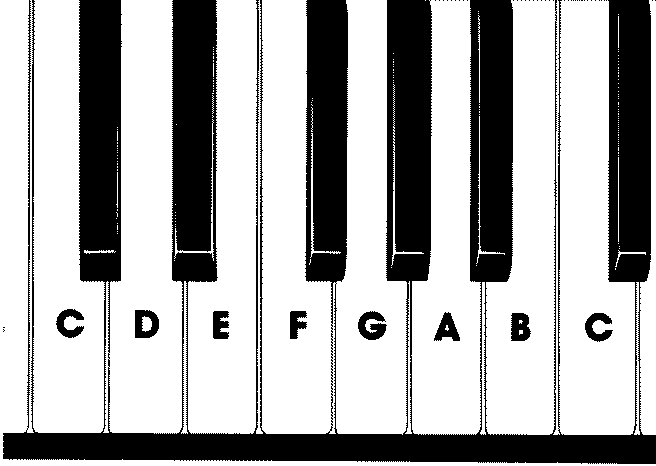
Introduction/Project Description

What is it?

Our Music Box project is a computer interfacing project. It is made up of thin pieces of wood glued together to make a rectangular prism. It is decorated with paint to look like a piano. With the use of a parallel port we can relay inputs from the switches to our program.

What does it do?

The goal of our music box is to play a complete octave (C D E F G A B C). Upon completing our goal we decided to add a customized music board to it. With the help of Turing we were able to play different sounds on the computer.



How does the user operate it?

The project consists of 3 main potions: the wiring, the parallel port and the program. For the wiring portion we have switches that acquire user input. The parallel port relays the input into the computer to the program. The program is the logic portion of the project and it also plays the sounds. I will explain more of these portions in detail later on in my report.

To split up our work I did the logic portion (programming) and Monil did the user input potion (wiring).

Theory of operation (How does it work electrically?)

The electrical portion of our project consists 8 major components: the switches, parallel port, buffer, power supply, a voltage regulator, diodes, resistors and conducting wires. We followed the schematic given and wired our circuit on a bread board and connected it to a parallel port.

Our switches are basic push buttons.

Our parallel port came in handy from our last project.

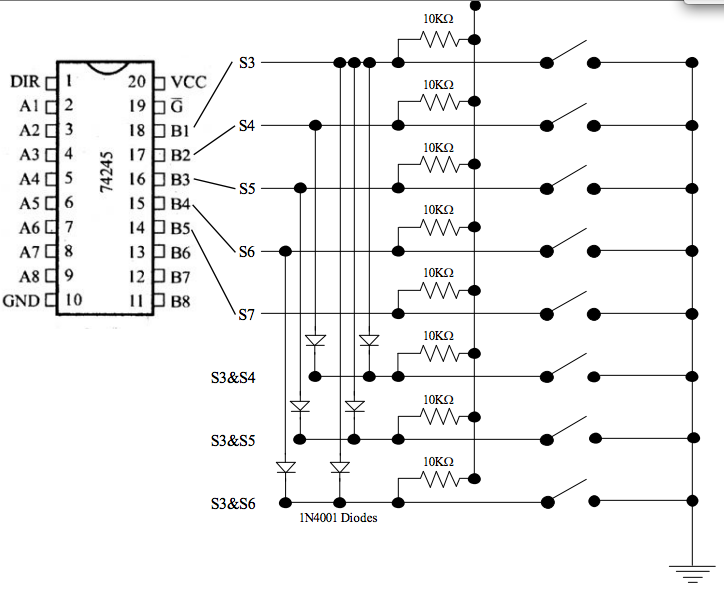
The buffer we used is the 74LS245.

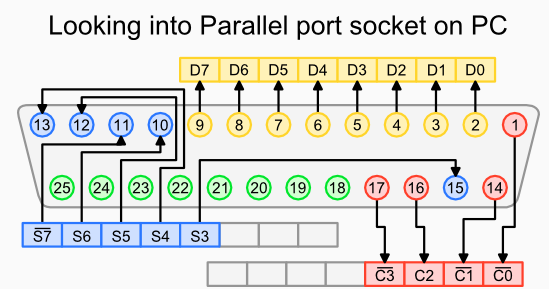
Our power supply is a 9V battery.

The voltage regulator is used to prevent damage to the buffer.

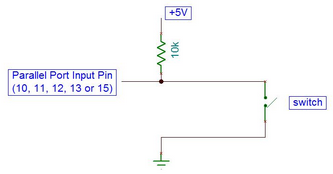
Polar diodes in our circuit are used when we need to get 2 inputs at the same time.

Resistors are used to lower the voltage going into the buffer.





To get the computer to read the switches we had to connect the switches to pins 10 (S6) , 11(S7), 12(S5), 13(S4), 15(S3).



This schematic shows that if there is a **high signal, aka 1, aka +5V**, to the computer the switches will be read as **OPEN**. When there is a **low signal, aka 0, aka GND**, to the computer the switches will be read as **CLOSED**. This is something we took into account whilst coding.

Construction Details and Results

With the lack of in class time we had due to countless excursions majority of our construction took place in school while the individual portions (wiring and programming) took place at home.

The materials used for the construction of our project were mainly leftovers that we found lying around. All we did was cut them up and shaped them to assemble a rectangular prism.

We had a total of 2 classes spent on troubleshooting and testing our work that we did at home. In the first class the wiring for a couple of switches were not done properly. In the second class we worked on construction for a while and tried the wiring again and we had no luck. Currently Monil is working on it at home and when it is due (June 4, 2015) we will finally be able to test it to see if it works or not.



Me working on construction at home

Me working on the construction of the house at home

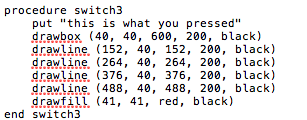
The problems we encountered were mainly timing issues. Since Monil and I both missed a lot of class time we were required to work on the project at home along with other classwork. This gave us a smaller time frame to work with. Although I personally didn’t have much time I managed to do what was required for the code and add an extra soundboard in it. I also assembled the project at home and painted it. At the end of the day we were able to finish before the due date.

Me working on the program at home

Code

Despite the lack of time we included graphics and another soundboard to the basic octave soundboard in our code.

To add graphics we made procedures for each scenario that display which switch has been pressed and every time the switch is pressed we simply called the procedure instead of typing it all out again. The commands we used were drawbox,: for the overall plane of the music board, drawline: to split each switch, and drawfill: to colour the switch being pressed. Here is an example of a procedure we made:



**Octave code listing:**

In order to code our octave we had to use the parallelget procedure to obtain the state of the switches. Like mentioned before when there is a low signal the switch will be close and the circuit will be complete.

How does parallelget work?

Parallelget uses the S3-S7 ports to obtain inputs. Each pin is assigned a value:

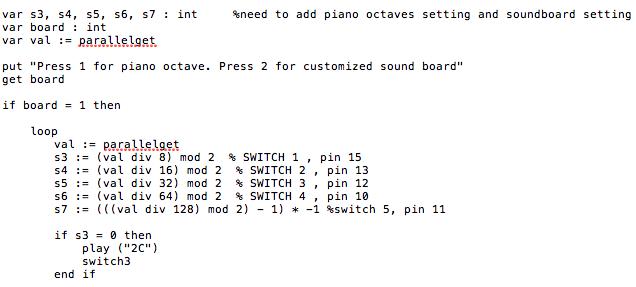
pin # on top and value on the bottom

To find the state of the switch you have to use 2 procedures, ‘div’ and ‘mod’. The div procedure divides a certain number (64, 128, 32, 16, 8 respectively) by the values to get an int number. The mod (modulo operator) finds the remainder so when you do anything mod 2 you get 1 or 0. An odd number mod 2= 1, and an even number mod 2=0. With this you can get the stage of the switches.

**S7 the special switch**:

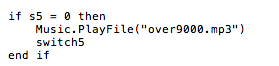
When you take the value of s7 and use div 128 you get an odd number. Therefore when you use mod on it you get 1 (high). To counteract this you need to do the following:

Where val is the value.

**Variable declaration and note example**In this code listing we named the switches and applied the div mod functions to get the state of the switches. In the segment that reads “if s3=0 then…” we added a play function that plays a half note C and prints the graphics out when s3 is pressed.

**Soundboard listing**:

For the soundboard portion we used the same logic as the octave listing but used a different command called Music.PlayFile.



This segment plays a predetermined file that we downloaded when s5 is pressed and it prints out the graphics of which switch is being pressed.

**RAW CODE**



Reflection

From this project I learned how to manage my time more wisely. With the lack of time I had to work with I had to compensate the workload by working at home. I also learned what an octave is and much more about music. As for turing I learned how to play music files and play notes. For the wiring I wasn’t appointed to do it but I feel like I had a good understanding of it before the project was assigned. I also learned about polar diodes and how you can tell which side is the cathode and which side is the anode.

I feel like Monil and I worked pretty well together to the job done. However we should improve next time by trying to get more troubleshoots and testing classes in.

The work was distributed like so:

Programming: me

Wiring: Monil

Construction: both

The logic and programming and everything on the computer was done by me. The wiring was all done by Monil. We both worked on the construction.