Midi Interface for Organ Pedal

Supervisor: Dr. Peter Driessen

Group #1

Bruce Lin (0120438)
Ronnie Lai (0121566)
Michael Lai (0330276)
# TABLE OF CONTENTS

LIST OF FIGURES AND TABLES ........................................................................................................ ii

FIGURES ........................................................................................................................................ ii

TABLES .......................................................................................................................................... ii

1.0 ABSTRACT .............................................................................................................................. 1

2.0 INTRODUCTION .................................................................................................................... 1

3.0 SYSTEM OVERVIEW ............................................................................................................. 1

3.1 Midi SPECIFICATION .............................................................................................................. 2

3.2 SWITCH CONNECTION ............................................................................................................ 4

3.3 HARDWARE .......................................................................................................................... 7

  3.3.1 Switches .......................................................................................................................... 7
  3.3.2 Multiplexers ..................................................................................................................... 8
  3.3.3 PIC18F2320 ................................................................................................................... 9

3.4 SOFTWARE DESIGN ............................................................................................................. 11

4.0 CONCLUSION ....................................................................................................................... 13

5.0 RECOMMENDATIONS ........................................................................................................... 13

6.0 APPENDIX

  Appendix A (Schematic) ......................................................................................................... 14
  Appendix B (General Midi Program 0-127) ............................................................................. 15
  Appendix C (Source Code) .................................................................................................... 17
LIST OF FIGURES AND TABLES

FIGURES

Figure 1: System Block Diagram of Midi Interface. ............................................................... 1
Figure 2: Three Enforce Chips. Three Enforce Chips. .......................................................... 4
Figure 3: The Built-in Switches. ....................................................................................... 4
Figure 4: The Output Connector. ...................................................................................... 5
Figure 5: Inside of the Female Connector. ....................................................................... 5
Figure 6: Labeled Wires. ................................................................................................. 6
Figure 7: The Overview of the Connector. ....................................................................... 6
Figure 8: Switch Connection (Circuit Diagram). ................................................................ 7
Figure 9: System Block Diagram of Hardware Connection ............................................... 8
Figure 10: Flow Chart of Software Implementation......................................................... 12

TABLES

Table 1: Channel Voice Message MIDI Table ................................................................. 3
Table 2: Connector Numbers. .......................................................................................... 5
Table 3: Summary table of PIC port A, B, and C ............................................................. 10
Table 4: Summary of Multiplexer Input Select ............................................................... 10
1.0 ABSTRACT

Many older and acoustic keyboard instruments do not have a MIDI interface and thus cannot be used to control other synthesizers. The purpose of the project is to design a midi interface that will allow the user to convert music notes to midi data. The midi interface is primarily used for organ pedal. Our goal is to build a wooden device that can be placed on top of an organ pedal (for portability reasons). When the user press the organ pedal, a sensor in the pedal detects that a key has been pressed. The sensor then transmits the data into the PIC microcontroller. The PIC collects these data, converts them to MIDI format and then transmits them to a MIDI port. These data can then be stored as a MIDI file (digital data) or output to the speaker for sound.

2.0 INTRODUCTION

An organ is a keyboard instrument that makes its sound by forcing air through wood or metal pipes. Most organ pipes are either long cylindrical metal tubes or elongated wooden boxes of rectangular cross-section and are most commonly found in churches, town halls, and art centres. Pedalboard is the name of a large keyboard at the base of an electronic or pipe organ console that the organist plays with his/her feet. In order to replace these organs with midi compatible pipe organs, it’ll cost thousands of dollars and a lot of work and time put into it. Due to the short amount of time, we have decided to make a midi interface only for the organ pedal that can translate the keynotes to midi information. If possible, we would like to make another midi interface for the whole set of organ pipe in the future. This report outlines the midi interface design and the efforts made to reduce old organ pipe replacement cost.

3.0 SYSTEM OVERVIEW

The midi interface is a device that translates midi signals from an electronic instrument into a form that can be read by a computer. A 32-switch connector is connected from the pedal board to our midi interface. When a keynote is pressed, the corresponded switch will be detected. This signal is then processed by a PIC (microcontroller) that translates the keynote to midi data, which can be send to and recorded by existing software such as Cakewalk, Noteworthy Composer or Maestro synthesizer on a computer through a midi cable. These notes are displayed on the computer screen when the recording is completed. The user can either save the midi file, transfer to another computer or replay within computer. An overall system setup is shown in figure 1.

![Figure 1: System Block Diagram of Midi Interface.](image-url)
The features of our midi compatible organ pedalboard are
- Standard 32 key (from C1 to G3),
- 32 key note on/off control with 32 LEDs,
- Fixed velocity to 127,
- Adjustable MIDI output channel from 1 to 16, and
- Program change increment/decrement of total of 128 programs.

3.1 Midi Specification

MIDI stands for Musical Instrument Digital Interface. The development of the MIDI system has been a major catalyst in the explosion of music technology. MIDI has put powerful computer instrument networks and software in the hands of less technically versed musicians and has provided new and time-saving tools for computer musicians.

The MIDI protocol is made up of messages. A message consists of a string in series of 8-bit bytes (0’s or 1’s). MIDI has many such defined messages. Some messages consist of only 1 byte. Other messages can have 2 or more bytes. The one thing that all messages have in common is that the first byte of the message is the Status byte. This is a special byte because it's the only byte that has bit #7 set. The bytes following the Status byte (called Data bytes) do not have bit #7 set. A start of a MIDI message is detected when bit #7 of the Status byte is set. Therefore, the Status byte ranges from 0x80 to 0xFF. The remaining bytes of the message will be in the range 0x00 to 0x7F.

The Status bytes of 0x80 to 0xEF are for messages that can be broadcast on any one of the 16 MIDI channels. Because of this, these are called Voice messages. For these Status bytes, you break up the 8-bit byte into two 4-bit nibbles. The high nibble specifies what type of MIDI message this is. Here are the possible values for the high nibble, and what type of Voice Category message each represents:

- 8 = Note Off
- 9 = Note On
- A = AfterTouch (i.e., key pressure)
- B = Control Change
- C = Program (patch) change
- D = Channel Pressure
- E = Pitch Wheel

The low nibble is the MIDI channel bit. There are 16 possible MIDI channels, with 0 being the first. Almost all MIDI devices are equipped to receive MIDI messages on one or more of 16 selectable MIDI channel numbers.
Table 1: Channel Voice Message MIDI Table

<table>
<thead>
<tr>
<th>Status Byte</th>
<th>Data Byte 1</th>
<th>Data Byte 2</th>
<th>Message</th>
<th>Legend</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000nnnn</td>
<td>0kkkkkkk</td>
<td>0vvvvvvv</td>
<td>Note Off</td>
<td>n=channel * k=key # 0-127 (60=middle C) v=velocity (0-127)</td>
</tr>
<tr>
<td>1001nnnn</td>
<td>0kkkkkkk</td>
<td>0vvvvvvv</td>
<td>Note On</td>
<td>n=channel k=key # 0-127 (60=middle C) v=velocity (0-127)</td>
</tr>
<tr>
<td>1010nnnn</td>
<td>0kkkkkkk</td>
<td>0pppppppp</td>
<td>Poly Key Pressure</td>
<td>n=channel k=key # 0-127 (60=middle C) p=pressure (0-127)</td>
</tr>
<tr>
<td>1011nnnn</td>
<td>0ccccccc</td>
<td>0vvvvvvv</td>
<td>Controller Change</td>
<td>n=channel c=controller v=controller value (0-127)</td>
</tr>
<tr>
<td>1100nnnn</td>
<td>0pppppppp</td>
<td>[none]</td>
<td>Program Change</td>
<td>n=channel p=preset number (0-127)</td>
</tr>
<tr>
<td>1101nnnn</td>
<td>0pppppppp</td>
<td>[none]</td>
<td>Channel Pressure</td>
<td>n=channel p=pressure (0-127)</td>
</tr>
<tr>
<td>1110nnnn</td>
<td>0ccccccc</td>
<td>0fffffff</td>
<td>Pitch Bend</td>
<td>n=channel c=coarse f=fine (c+f = 14-bit resolution)</td>
</tr>
</tbody>
</table>

For example, a message for turning on a note (middle C) on MIDI channel #5 very loudly (with a velocity or force of 127, the maximum) is shown below in binary.

<table>
<thead>
<tr>
<th>status byte</th>
<th>data byte</th>
<th>data byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>10010100</td>
<td>00111100</td>
<td>01111111</td>
</tr>
</tbody>
</table>

Simultaneous events in MIDI must be sent as a string of serial commands. A 3-note chord, for example, will be transmitted as three separate note #-velocity pairs. Because of the 31.25 Kbaud transmission speed, this is normally perceived as simultaneity. To make more efficient use of the limited bandwidth, MIDI manufacturers adopted a shortcut called running status. Running status allows a single status byte's action to remain in effect for an unlimited number of data byte pairs which follow. For example, to play three 'simultaneous' notes on the same MIDI channel, a Note On status byte can be sent, followed by six data bytes.

**status(note on, ch 1) key1-velocity key2-velocity key3-velocity**

To help minimize excessive data by using running status, the Note On command can also function to turn notes off by sending a velocity value of zero for the key # to be turned off.

**status(note on, ch 1) key1-velocity1 key2-velocity2 key3-velocity3 key1-velocity0 key2-velocity0 key3-velocity0**
3.2 Switch Connections

For this particular organ pedal, there is one built in switch for each pedal. When the pedal is pressed, the switch will be closed. When the foot is released, the enforce chip will push the pedal back. The switch will disconnect the two points to form an open circuit. The enforce chip is shown in the figure 2 as below.

![Figure 2: Three Enforce Chips](image)

The built-in switch contains a piece of metal and four pins. When the pedal is pressed, this metal will connect these four pins. The metal and the four pins are shown in the figure 3. One of the four pins functions as a reference point. The other three pins are connected to the three different output connectors, and their function is the same. One of the three output connectors are shown in figure 4. When the pedal is pressed, the metal will connect these three pins to the reference point (which is the ground). The reason for three pins is that in case one pin does not conduct with the metal, hopefully, the other two pins will conduct. This ensures better conductivity.

![Figure 3: The Built-in Switches](image)
After some researches were done, the output connector is identified as 33-COND MT. 300 SERIES. The female of the connector is ordered from the website: http://www.electronicsurplus.com/. The inside of the female connector is painted black. In order to improve the conductivity of the wire, the inside of the connector is sanded and the wires are soldered onto them. Please see the figure 5 for the detail of the connections. The other side of the wire is labeled so we know which one corresponds to which switch (figure 6). Finally, the overview of the connector is shown in figure 7.

<table>
<thead>
<tr>
<th>Table 2: Connection Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Figure 4: The Output Connector

Figure 5: Inside of the Female Connector
The switch component discussed here is different from the progress report 2. The reason is that built in switches are found on the pedalboard. However, the previous design is still feasible however it will be more complicated than the one discussed in this section to build. For the pedalboard which doesn’t have the built-in switches, the design discussed in progress report 2 is an excellent choice.
3.3 HARDWARE

3.3.1 Switches

**Objective:** the objective of the switches is to output low when a pedal is being pushed and output high when a pedal is released or not pushed.

**Explanation:** the switch is simply made of normally open push button. Upon further investigation of the pedalboard, we found that there is a built in switch for each pedal (32 in total), so there was no need to construct build an external device to mount onto the pedalboard. Instead, a connector was purchased and connected to the built in connector on the pedalboard.

![Switch Connection (Circuit Diagram)](image)

5V supply, resistors and LEDs are all built on the midi interface board. LED is used to display if a pedal is being pushed.

Here are 8 sample switches built in the pedalboard. It is similar to a normally open push button switch.

**Figure 8: Switch Connection (Circuit Diagram)**
3.3.2 Multiplexers

**Objective:** the objective of the multiplexer is to retrieve 32 pedal notes and transfer them to the PIC microcontroller.

**Explanation:** although PIC18F2320 supports 3 ports (24 Input/Output), Port C is reserved for crystal oscillator and SPI communication. So that leaves 16 input/output available. Moreover, 3 outputs are reserved to control the multiplexer input select. Therefore this is not enough to retrieve all 32 signals from the pedalboard. As a result, four 8 to 1 multiplexers were used.

The first multiplexer is responsible for the 1st 8 switches on the pedalboard. The second multiplexer is responsible for the next 8 switches.

**Figure 9: System block diagram of hardware connection.**

![Diagram of hardware connection](image-url)
3.3.3 PIC18F2320

**Objective:** The objective of the PIC is to retrieve key note data from the pedal organ, convert them to MIDI data bytes and transmit the data to a MIDI port via serial peripheral interface.

**Explanation:** The PIC is responsible to control the multiplexers. When an input is selected from the multiplexer, the PIC has to know which input from the multiplexer corresponds to which key note from the pedalboard. These data must be stored in the PIC and transferred to a MIDI port via SPI.

Port RB0, RB1, RB2 and RB3 are set as input. These ports are used to retrieve signals from the multiplexer. When the signal is received, it is immediately stored in an array.

Port RA0, RA1, and RA2 are used for multiplexer select. Multiplexer select is used to select one of the 8 multiplexer inputs. This method allows us to use one output for 8 signals.

Port RA3 is used to enable the multiplexer. When set to low, multiplexer is activated.

Port RA4 and RA5 are used to select the types of instrument to play. The default is set to Church Organ as it is a pedalboard after all. The instrument select is just an additional thing that we added into the circuit for fun. Since MIDI allows us to play up to 128 different instruments, we thought it'll be fun and will be a good presentation on the presentation day.

Port RA6 is used to select bass or middle C. Since keeping the setting at bass when we change the type of instrument to play will not sound very pleasant. As a result, this switch is implemented so that if the user would like to use the pedalboard as organ bass, he/she can set it by pressing the button connected to Port RA6 (shown on circuit schematic). If the user leaves as default, he/she can play middle C notes.

After all 32 signals from the organ pedal is retrieved and stored. The PIC converts these data to digital signals in MIDI data format. The data is then transmitted to a MIDI port through SPI.

Port RC6 is used for sending data to a MIDI port serially.
Table 3: Summary Table of PIC Port A, B and C

<table>
<thead>
<tr>
<th>Port</th>
<th>Pin Name</th>
<th>Input/Output</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>RA0 – RA2</td>
<td>Output</td>
<td>Multiplexer Select Input (S0, S1, S2)</td>
</tr>
<tr>
<td></td>
<td>RA3</td>
<td>Output</td>
<td>Multiplexer enable Input (active low E)</td>
</tr>
<tr>
<td></td>
<td>RA4</td>
<td>Input</td>
<td>Change program (decrease)</td>
</tr>
<tr>
<td></td>
<td>RA5</td>
<td>Input</td>
<td>Change program (increase)</td>
</tr>
<tr>
<td></td>
<td>RA6</td>
<td>Input</td>
<td>Change to bass or middle C</td>
</tr>
<tr>
<td>B</td>
<td>RB0 – RB3</td>
<td>Input</td>
<td>Multiplexer output to PIC</td>
</tr>
<tr>
<td>C</td>
<td>RC6</td>
<td>Output</td>
<td>SPI TX to cable</td>
</tr>
</tbody>
</table>

Table 3: Summary of Multiplexer Input Select

<table>
<thead>
<tr>
<th>Select Input (S2,S1,S0)</th>
<th>PIC Input (key notes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RB3</td>
</tr>
<tr>
<td>0 0 0</td>
<td>25</td>
</tr>
<tr>
<td>0 0 1</td>
<td>26</td>
</tr>
<tr>
<td>0 1 0</td>
<td>27</td>
</tr>
<tr>
<td>0 1 1</td>
<td>28</td>
</tr>
<tr>
<td>1 0 0</td>
<td>29</td>
</tr>
<tr>
<td>1 0 1</td>
<td>30</td>
</tr>
<tr>
<td>1 1 0</td>
<td>31</td>
</tr>
<tr>
<td>1 1 1</td>
<td>32</td>
</tr>
</tbody>
</table>

Multiplexer 1 is responsible for switches 1-8 as you can see on the table port RB0. Multiplexer 2 is responsible for the next set of switches, then multiplexer 3...

As you can see from the table, 8 switches share a multiplexer output. This allows us to use four bits of port B to retrieve 32 signals.

These data are stored in an array in the following order:
1,9,17,25,2,10,18,26,3,11,19,27,4,12,20,28,5,13,21,29,6,14,22,30,7,15,23,31,8,16,24,32
3.4 SOFTWARE DESIGN

Introduction

The only software development in this project is programming the PIC microcontroller. There are 3 sections to the software development. The first part is to detect the switches and store the data on an array. The second part is to convert these data into midi data bytes. The third part is to send all the data stored in an array via Serial Peripheral Interface. The code is attached in appendix C. Figure 10 shows the flow chart of the software implementation.

1. Storing the data on an array
To store data on an array, we simply create a for loop and retrieve signals from each input of the multiplexer. The for loop increments the multiplexer input select and then read RB0 – RB3. The read data is then stored in an array. The stored array is compared with previous stored array to check which keys have been pressed or released.

2. Converting to MIDI data byte
MIDI port requires that a 2 bytes or 3 bytes be sent to it. A MIDI byte consists of a STATUS byte and a DATA byte.

The first bit in the STATUS byte is always used to indicate whether byte is STATUS or DATA. The lowest four bits are used for midi channel. The DATA byte is used for actual data. It is the key NOTE.

In this project, only three MIDI messages are considered. They are Note Off, Note On, and Program Change. Total of 3 bytes was translated from received signals for Note off and Note On, while only 2 bytes was translated for Program Change. The first byte is STATUS byte which tells what the message is and which channel is selected. The second byte is DATA byte that specifies the music key and program number for Note Off/On message and Program Change message respectively. The third extra byte that Note Off/On has is another DATA byte that fixes the velocity to maximum 127.

More on MIDI specification is in MIDI specification section.

3. Transferring data via SPI
To transfer data to a MIDI port, we use the function putc() in C compiler. The function is already setup for SPI transfer. The byte placed in the function is automatically sent through SPI to the MIDI port.
Figure 10: Flow Chart of Software Implementation

Start

Enable Multiplexer

True

i <= 7

Retrieve data from multiplexer and store in array

True

False

k <= 31

Output data stored in array to SPI Tx

End
4.0 CONCLUSION

The Midi interface is successfully connected to the outdate organ pedalboard. When the key from the organ pedal is pressed, the corresponding signal is transferred from the midi interface to the computer. As a result, the particular tone will sound, and can be recorded in real-time. The cost for the portable midi interface is less than 30 dollars which is affordable to everyone. All in all, this project design course was successfully completed.

5.0 RECOMMENDATIONS

There are several things that can be improved on the Midi interface. First, since there are 128 programs to choose from, it will be much easier to have a digital display than to guess which instrument is playing. Secondly, there are only an increment switch and a decrement switch to change the midi program. It might be easier to change the Midi program with a numeric keypad. Thirdly, the built in switch in the pedalboard has a lot of rust. The performance can be improved by removing the rust spots. Finally, another switch can be added into the pedalboard to detect velocity. This will allow for richer, more natural sound instead of the sound generated fixed at maximum when a key is pressed.
### Appendix B – General Midi Programs (0-127)

<table>
<thead>
<tr>
<th>Number</th>
<th>Program Name</th>
<th>Number</th>
<th>Program Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Acoustic Grand Piano</td>
<td>64</td>
<td>Soprano Sax</td>
</tr>
<tr>
<td>1</td>
<td>Bright Acoustic Piano</td>
<td>65</td>
<td>Alto Sax</td>
</tr>
<tr>
<td>2</td>
<td>Electric Grand Piano</td>
<td>66</td>
<td>Tenor Sax</td>
</tr>
<tr>
<td>3</td>
<td>Honky-tonk Piano</td>
<td>67</td>
<td>Baritone Sax</td>
</tr>
<tr>
<td>4</td>
<td>Electric Piano 1</td>
<td>68</td>
<td>Oboe</td>
</tr>
<tr>
<td>5</td>
<td>Electric Piano 2</td>
<td>69</td>
<td>English Horn</td>
</tr>
<tr>
<td>6</td>
<td>Harpsichord</td>
<td>70</td>
<td>Bassoon</td>
</tr>
<tr>
<td>7</td>
<td>Clavichord</td>
<td>71</td>
<td>Clarinet</td>
</tr>
<tr>
<td>8</td>
<td>Celesta</td>
<td>72</td>
<td>Piccolo</td>
</tr>
<tr>
<td>9</td>
<td>Glockenspiel</td>
<td>73</td>
<td>Flute</td>
</tr>
<tr>
<td>10</td>
<td>Music Box</td>
<td>74</td>
<td>Recorder</td>
</tr>
<tr>
<td>11</td>
<td>Vibraphone</td>
<td>75</td>
<td>Pan Flute</td>
</tr>
<tr>
<td>12</td>
<td>Marimba</td>
<td>76</td>
<td>Blown Bottle</td>
</tr>
<tr>
<td>13</td>
<td>Xylophone</td>
<td>77</td>
<td>Shakuhachi</td>
</tr>
<tr>
<td>14</td>
<td>Tubular Bells</td>
<td>78</td>
<td>Whistle</td>
</tr>
<tr>
<td>15</td>
<td>Dulcimer</td>
<td>79</td>
<td>Ocarina</td>
</tr>
<tr>
<td>16</td>
<td>Drawbar Organ</td>
<td>80</td>
<td>Lead 1 (square)</td>
</tr>
<tr>
<td>17</td>
<td>Percussive Organ</td>
<td>81</td>
<td>Lead 2 (sawtooth)</td>
</tr>
<tr>
<td>18</td>
<td>Rock Organ</td>
<td>82</td>
<td>Lead 3 (calliope)</td>
</tr>
<tr>
<td>19</td>
<td>Church Organ</td>
<td>83</td>
<td>Lead 4 (chiff)</td>
</tr>
<tr>
<td>20</td>
<td>Reed Organ</td>
<td>84</td>
<td>Lead 5 (charang)</td>
</tr>
<tr>
<td>21</td>
<td>Accordion</td>
<td>85</td>
<td>Lead 6 (voice)</td>
</tr>
<tr>
<td>22</td>
<td>Harmonica</td>
<td>86</td>
<td>Lead 7 (fifths)</td>
</tr>
<tr>
<td>23</td>
<td>Tango Accordion</td>
<td>87</td>
<td>Lead 8 (bass + lead)</td>
</tr>
<tr>
<td>24</td>
<td>Acoustic Guitar</td>
<td>88</td>
<td>Pad 1 (new age)</td>
</tr>
<tr>
<td>25</td>
<td>Acoustic Guitar</td>
<td>89</td>
<td>Pad 2 (warm)</td>
</tr>
<tr>
<td>26</td>
<td>Electric Guitar</td>
<td>90</td>
<td>Pad 3 (polysynth)</td>
</tr>
<tr>
<td>27</td>
<td>Electric Guitar</td>
<td>91</td>
<td>Pad 4 (choir)</td>
</tr>
<tr>
<td>28</td>
<td>Electric Guitar</td>
<td>92</td>
<td>Pad 5 (bowed)</td>
</tr>
<tr>
<td>29</td>
<td>Overdriven Guitar</td>
<td>93</td>
<td>Pad 6 (metallic)</td>
</tr>
<tr>
<td>30</td>
<td>Distortion Guitar</td>
<td>94</td>
<td>Pad 7 (halo)</td>
</tr>
<tr>
<td>31</td>
<td>Guitar harmonics</td>
<td>95</td>
<td>Pad 8 (sweep)</td>
</tr>
<tr>
<td>32</td>
<td>Acoustic Bass</td>
<td>96</td>
<td>FX 1 (rain)</td>
</tr>
<tr>
<td>33</td>
<td>Electric Bass (finger)</td>
<td>97</td>
<td>FX 2 (soundtrack)</td>
</tr>
<tr>
<td>34</td>
<td>Electric Bass (pick)</td>
<td>98</td>
<td>FX 3 (crystal)</td>
</tr>
<tr>
<td>35</td>
<td>Fretless Bass</td>
<td>99</td>
<td>FX 4 (atmosphere)</td>
</tr>
<tr>
<td>36</td>
<td>Slap Bass 1</td>
<td>100</td>
<td>5 (brightness)</td>
</tr>
<tr>
<td>37</td>
<td>Slap Bass 2</td>
<td>101</td>
<td>FX 6 (goblins)</td>
</tr>
<tr>
<td>38</td>
<td>Synth Bass 1</td>
<td>102</td>
<td>FX 7 (echoes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>39 Synth Bass 2</td>
<td>103 FX 8 (sci-fi)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40 Violin</td>
<td>104 Sitar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>41 Viola</td>
<td>105 Banjo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>42 Cello</td>
<td>106 Shamisen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43 Contrabass</td>
<td>107 Koto</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44 Tremolo Strings</td>
<td>108 Kalimba</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45 Pizzicato Strings</td>
<td>109 Bag pipe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>46 Orchestral Harp</td>
<td>110 Fiddle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>47 Timpani</td>
<td>111 Shanai</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48 String Ensemble 1</td>
<td>112 Tinkle Bell</td>
<td></td>
<td></td>
</tr>
<tr>
<td>49 String Ensemble 2</td>
<td>113 Agogo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50 SynthStrings 1</td>
<td>114 Steel Drums</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51 SynthStrings 2</td>
<td>115 Woodblock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>52 Choir Aahs</td>
<td>116 Taiko Drum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>53 Voice Oohs</td>
<td>117 Melodic Tom</td>
<td></td>
<td></td>
</tr>
<tr>
<td>54 Synth Voice</td>
<td>118 Synth Drum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>55 Orchestra Hit</td>
<td>119 Reverse Cymbal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56 Trumpet</td>
<td>120 Guitar Fret Noise</td>
<td></td>
<td></td>
</tr>
<tr>
<td>57 Trombone</td>
<td>121 Breath Noise</td>
<td></td>
<td></td>
</tr>
<tr>
<td>58 Tuba</td>
<td>122 Seashore</td>
<td></td>
<td></td>
</tr>
<tr>
<td>59 Muted Trumpet</td>
<td>123 Bird Tweet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60 French Horn</td>
<td>124 Telephone Ring</td>
<td></td>
<td></td>
</tr>
<tr>
<td>61 Brass Section</td>
<td>125 Helicopter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>62 SynthBrass 1</td>
<td>126 Applause</td>
<td></td>
<td></td>
</tr>
<tr>
<td>63 SynthBrass 2</td>
<td>127 Gunshot</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix C – Source Code

// miditech.c
// Programmer: Michael Lai
// Date Modified: July 17, 2005
// Objective: To receive signals from Organ Pedal and convert to MIDI

#include <18f2320.h>
#include <midi.h>

#define NUM_INPUTS 32 // total number of keys
#define VELCRYSTAL 127 // max volume
#define CHURCH_ORGAN 19 // church organ program
#define REED_ORGAN 20 // read organ program

byte data_new[NUM_INPUTS]={0};
byte data_old[NUM_INPUTS]={0};

// key for organ base
byte key_note_base[NUM_INPUTS]={0,8,16,24,1,9,17,25,2,10,18,26,3,
                                11,19,27,4,12,20,28,5,13,21,29,6,14,22,30,7,15,23,31};

//byte key_note[NUM_INPUTS]={24,32,40,48,25,33,41,49,26,34,42,50,27,35,43,51,
                      //   28,36,44,52,29,37,45,53,30,38,46,54,31,39,47,55};

// key for middle C
byte key_note_mc[NUM_INPUTS]=
  {48,56,64,72,49,57,65,73,50,58,66,74,51,59,67,75
    52,60,68,76,53,61,69,77,54,62,70,78,55,63,71,79};

void main(){
    int i; // counter to control multiplexer
    int j; // counter to store each input in data
    int k; // counter to output all 32 keys to SPI
    byte instrument = CHURCH_ORGAN; // default to Church Organ
    byte instrument_old = CHURCH_ORGAN;
    byte inc;
    byte dec;
    byte key_note; // bass or middle C. Base for organ pedal. Middle C
    for piano...
    byte key_note_new; // check if button for change from organ pedal or middle C is
                        // being pressed
    // enable multiplexer
    output_low(PIN_A3);
key_note = 1;  // initialize to middle c

// change midi program to Church Organ
putc(kPgmChng);
putc(instrument);

while(TRUE){
    j=0;

    // check if program changed
    if (instrument_old != instrument){
        putc(kPgmChng);
        putc(instrument);
        instrument_old = instrument;
    }

    for (i=0;i<=7;i++){
        // multiplexer select
        output_a(i);
        delay_us(1);

        // retrieve data from each input
        data_new[j++] = input(PIN_B0);
        data_new[j++] = input(PIN_B1);
        data_new[j++] = input(PIN_B2);
        data_new[j++] = input(PIN_B3);
    }

    // output to SPI if there is a change in the pedal
    for (k=0; k<=31; k++){  
        if (data_new[k]!=data_old[k]){
            if (key_note){
                if (data_new[k]){
                    putc(kNoteOn);
                    putc(key_note_mc[k]);
                    putc(velocity);
                }
                else{
                    putc(kNoteOff);
                    putc(key_note_base[k]);
                    putc(velocity);
                }
            } else{
                if (data_new[k]){
                    putc(kNoteOn);
                    putc(key_note_base[k]);
                    putc(velocity);
                } else{
                    if (key_note){
                        putc(kNoteOff);
                        putc(key_note_base[k]);
                        putc(velocity);
                    } else{
                        putc(kNoteOff);
                        putc(key_note_base[k]);
                        putc(velocity);
                    }
                }
            }
        }
    }
}
} // end if
// copies new data to old
data_old[k] = data_new[k];
}
} // end for loop

inc = input(PIN_B5);
dec = input(PIN_B4);

// change instrument
if (!inc){
delay_ms(200);
instrument = (instrument+1)%128;
}
if (!inc){
delay_ms(200);
if (instrument == 0)
instrument = 127;
else
instrument = instrument-1;
}

// see if switch is in the Organ pedal or other position
key_note_new = input(PIN_B6);

// change from base to middle c or vice versa
if (!key_note_new){
if (key_note == 1)
key_note = 0;
else
key_note = 1;
delay_ms(200);
}

delay_ms(5);

}//end while loop

}//end of program