Guide to DVD Chapter 15 Examples: *Tim Lukens*

The MIDI Spec and Programming With PortMIDI

Setting up the IAC as a MIDI Device (OS X)

The examples in this chapter all require a valid MIDI device. There is a simple way inside of OS X to use the IAC to control softsynths on your computer.

1. Use Spotlight to open Audio MIDI Setup.
2. Click the MIDI devices tab.
3. Double click on IAC Driver.
4. Make sure the option “Device is Online” is checked.
5. If the box labeled “Ports” is empty, click the plus button beneath it to add a new IAC port.
6. Open your softsynth of choice and use the options to select IAC Bus 1 as the MIDI input device.
7. When running any of the following examples, use IAC Bus 1 as the MIDI output device.

Devices

This program prints all of the MIDI devices connected to the computer. The command to compile is as follows:

```
$ gcc -o devices devices.c -lportmidi -framework coremidi
```

If this is not being compiled in OS X, the `-framework coremidi` can be dropped. Running this program is simple:

```
$ ./devices
```

You should expect to see either a list of currently connected MIDI devices or an error stating there are no MIDI devices plugged in to your computer.

Devices With Stream

This program is the same as *devices.c* with the added feature of opening a stream to a MIDI device. The command to compile is as follows:

```
$ gcc -o devicesWithStream devicesWithStream.c -lportmidi -framework coremidi
```

If this is not being compiled in OS X, the `-framework coremidi` can be dropped. Running the program is as follows:

```
$ ./devicesWithStream
```

This program should run exactly the same as *devices.c*.

MIDI Read

This program reads MIDI messages from a controller and prints them to the terminal. This program will only listen on MIDI channel 1. The command to compile is as follows:

```
$ gcc -o midiread midiread.c -lportmidi -framework coremidi
```

If this is not being compiled in OS X, the `-framework coremidi` can be dropped. Running is as follows:

```
$ ./midiread
```

This program requires a MIDI device to be plugged into your computer. Playing notes or changing CC values will print the status and data bytes to the screen. Use ctrl-c to quit the program.

Auto Assigning Controllers

This program allows the user to select which MIDI controller to use by moving the desired controller. This program will only listen on MIDI channel 1. The command to compile is as follows:

```
$ gcc -o autoassign autoassign.c -lportmidi -framework coremidi
```

If this is not being compiled in OS X, the `-framework coremidi` can be dropped. Running is as follows:

```
$ ./autoassign
```

This program requires a MIDI device to be plugged into your computer. The program will ask the user to move a MIDI controller. When you do, the program will print out which controller was selected and end. If no controller is readable, the program can be quit by using ctrl-c.

Auto Assigning Continuous Controllers

This program allows the user to choose their CCs for different aspects of the program by moving them when asked. This program will only listen on MIDI channel 1. The command to compile is as follows:

```
$ gcc -o autoassigncc autoassigncc.c -lportmidi -framework coremidi
```

If this is not being compiled in OS X, the `-framework coremidi` can be dropped. Running is as follows:

```
$ ./autoassigncc
```

This program runs exactly the same as *autoassign.c* except it then continues on to ask the user to move a few different CCs to store them to a matrix. The program then prints out the new CC matrix and ends.

Using the Buffer

This program demonstrates how to take advantage of the PortMIDI buffer. This program will only listen on MIDI channel 1. The command to compile is as follows:

```
$ gcc -o buffer buffer.c -lportmidi -framework coremidi
```

If this is not being compiled in OS X, the `-framework coremidi` can be dropped. Running is as follows:

```
$ ./buffer
```

This program uses code from *autoassign.c* to have the user select a MIDI device. It then uses the buffer to display MIDI messages to the screen. The program can be exited by using Ctrl-c.

MIDI Output

This program demonstrates how to pack MIDI messages and send them to an output stream. The command to compile is as follows:

```
$ gcc -o midioutput midioutput.c -lportmidi -framework coremidi
```

If this is not being compiled in OS X, the `-framework coremidi` can be dropped. Running is as follows:

```
$ ./midioutput
```

This program starts by printing all available MIDI output devices and asking the user to select one by entering the device number. This is where using the IAC will come in handy (see the beginning of the Examples Guide). After a MIDI output device is selected, the program will send twenty notes to the stream and quit.

Algorithmic Fifths

This program algorithmically generates MIDI sequences based on a simple scale and perfect fifths. The messages are then packed and sent to a MIDI output stream. The command to compile is as follows:

```
$ gcc -o algoFifths algoFifths.c -lportmidi -framework coremidi
```

If this is not being compiled in OS X, the `-framework coremidi` can be dropped. Running is as follows:

```
$ ./algoFifths
```

This program runs similarly to *midioutput.c* except it will algorithmically decide what to play. The program plays sequences of notes in the same pattern every time: A root note, followed by fifteen notes a perfect fifth above the root note. The terminal window will display the time between notes as well as the time between sequences. This program can create some interesting, ambient textures if multiple instances of the program are run at the same time. The IAC can be used to route to different synthesizers on your computer (use the plus button in Audio MIDI Setup to create more busses). Use Ctrl-c to quit the program.

Arpeggiator

This program reads MIDI from a stream and outputs the data in an arpeggiated fashion. This program will only listen on MIDI channel 1. The command to compile is as follows:

```
$ gcc -o arp arp.c -lportmidi -framework coremidi
```

If this is not being compiled in OS X, the `-framework coremidi` can be dropped. Running is as follows:

```
$ ./arp
```

This program requires both a MIDI controller and some sort of output. The IAC is always a valid output option on OS X. This program will read in one note at a time from a MIDI controller and output a series of notes to the output stream. This program can be quit by using Ctrl-c.

The Beat Cutter

This program reads an uncompressed (.wav, .aif, .aiff) audio file (mono or stereo) and slices it into even slices. The slices can then be played forwards or in a random order. The slice length, slice holds, direction, and enveloping technique can be controlled in real time with a MIDI controller. NOTE: The cutter will only listen on MIDI channel 1. The command to compile is as follows:

```
$ gcc cutter.c -o cutter -lportmidi -lportaudio lib/libportsf.a lib/portsf.a
-framework coremidi -framework coreaudio
```

If this is not being compiled in OS X, the `-framework coremidi` and `-framework coreaudio` can be dropped. This program has a few options when being run:

```
$ ./cutter audiofile.wav [slices, holds, random(1 or 0), MIDI(1 or 0)]
```

The only required argument is the audio file. The rest of the arguments are optional:

`slices` determines the initial number of slices the audio file should be cut into (1, 2, 4, 8, 16, 32, etc.).
`holds` determines the initial number of times to repeat a slice (1-15). `random` determines whether or not the file plays through the slices in order (0) or in a random order (1).
`MIDI` determines whether or not a MIDI controller is to be used to change these parameters in real-time.

Here is an example of a way to run the program:

```
$ ./cutter drumloop.wav 16 2 0 1
```

Once the program is running, if MIDI is turned on, the user can move the assigned CCs to control the number of slices, the number of slice holds, random on or off, and the slice envelope type.