The Complete Guide to SunVox

Release 0.1.0

Matthew Scott

Nov 07, 2017

Contents

Note:

[TBW] To be written.

[WIP] Writing in progress.

CHAPTER 1

Preface [WIP]

SunVox is a portable music studio that fits perfectly into my life. It might also fit well into yours, as it has for thousands of others.

I discovered SunVox in early 2016, after a 12-year break from writing music. I was looking for ways to get back into it with modern tools, but without spending a lot of money.

"What a great tool this is," I thought to myself after much experimentation. I was already convinced that SunVox was a tool I was going to spend many hours with.

Then, I discovered that it was also avalable for my phone. Everything I loved about SunVox now fit into my pocket!

A year later, I find myself writing this preface to this book, and actively maintaining other projects that aim to help people get the most out of SunVox.

Several people from the international SunVox community helped with the creation of this book, and they are all listed at the *List of contributors* page.

This book is our gift to you, the reader. Whatever your skill level, we hope you will find delight in learning about SunVox!

– Matthew R. Scott, Editor

Contributors to this page

Author(s) Matthew Scott

CHAPTER 2

Introduction [WIP]

2.1 Who is this book for?

This book is for anyone interested in SunVox, the music studio app you can use on almost any computer.

Here are some of the types of people who might enjoy this book:

Beginner musicians Learn about music by using SunVox to compose songs using simple techniques. As you progress, you can add more layers and sounds to your projects.

Professional musicians Use SunVox by itself to create your next masterpiece, or connect it to the rest of your music studio.

Sound designers Create and discover exciting new timbres using SunVox's powerful audio engine. Complex synths and effects can be bundled into easy-to-use *MetaModules*, making them easy to organize and easy to share with others.

Developers of apps and games Embed SunVox music and sound effects into your next game using the royalty-free SunVox DLL. Develop

Chiptune and tracker enthusiasts If you already know what trackers are, and like the sound of chiptunes, enjoy SunVox's blend of a familiar tracker interface and modern synthesis tools.

2.2 What is SunVox?

From the SunVox home page:

> SunVox is a small, fast and powerful modular synthesizer with pattern-based sequencer (tracker). It is a tool for those people who like to compose music wherever they are, whenever they wish. On any device. SunVox is available for Windows, OS X, Linux, Maemo, Meego, Raspberry Pi, Windows Mobile (WindowsCE), PalmOS, iOS and Android.

2.3 What is a modular synthesizer?

Synthesizers, or *synths*, are electronic instruments that receives musical notes and other commands, transform them into sounds, and send out audio signals.

SunVox is a *modular synthesizer*, a synth that is composed of smaller components that can be easily *patched* (connected and disconnected from each other) to create complex sounds.

(In contrast, non-modular synths usually have parameters you can change, but do not allow rearranging their internal structures.)

2.4 What is a pattern-based sequencer?

Sequencers send a pre-programmed notes and other commands to synthesizers. The notes and commands are sent at precise times, following the tempo of the music being played.

SunVox is a *pattern-based sequencer*, which lets you create groups of these notes and commands. Each of these groups is called a *pattern*. This makes it easy to rearrange music, and to repeat sections more than once.

2.5 What is a tracker?

SunVox offers a *tracker* interface, which is a way of programming sequencer patterns by placing notes and commands into a grid of *tracks* and *lines*.

Tracks are arranged horizontally, and lines are arranged vertically. As a pattern plays, the sequencer steps through each line in the pattern, following the current tempo.

The currently-playing line of a pattern is then scanned for notes and commands in each track. Everything on that line is then sent to synthesizer modules, and played back at the same time.

2.6 What makes SunVox unique compared to similar apps?

- SunVox is one of the lowest-priced modular synths on the market. It's free on most supported platforms, and very affordable on others.
- The powerful *timeline* interface used to arrange patterns is unlike any other
- You can create, share, and modify complex module setups using *MetaModules*.
- It uses the same interface on all platforms. Learn SunVox concepts once, then use the same and techniques and project files on any supported platform.
- For developers, a royalty-free DLL is available. You can embed SunVox music and sound effects into your app.

2.7 How to read this book

Beginners may want to read the book sequentially, to build up an understanding of basic concepts before moving to more advanced topics.

Intermediate and expert-level SunVox users may want to skip some of the basics and jump to more advanced topics right away, or use this book as a reference manual.

To help you get the most from the book regardless of how you read it, we'll offer images, animated GIFs, diagrams, links to other sections, audio samples, and downloadable SunVox projects.

2.8 Conventions used in this book

(to be written)

Contributors to this page

Author(s) Matthew Scott

CHAPTER 3

Getting Started [TBW]

3.1 Installing SunVox [TBW]

3.1.1 Downloading

3.1.2 Installing

Android [TBW]

iOS [TBW]

Linux [TBW]

macOS [TBW]

Maemo [TBW]

Meego [TBW]

PalmOS [TBW]

Raspberry Pi [TBW]

Windows [TBW]

Windows CE [TBW]

3.2 Choosing a color theme [WIP]

When you start SunVox the first time, you are asked to choose a color theme. Select a color scheme and click OK.

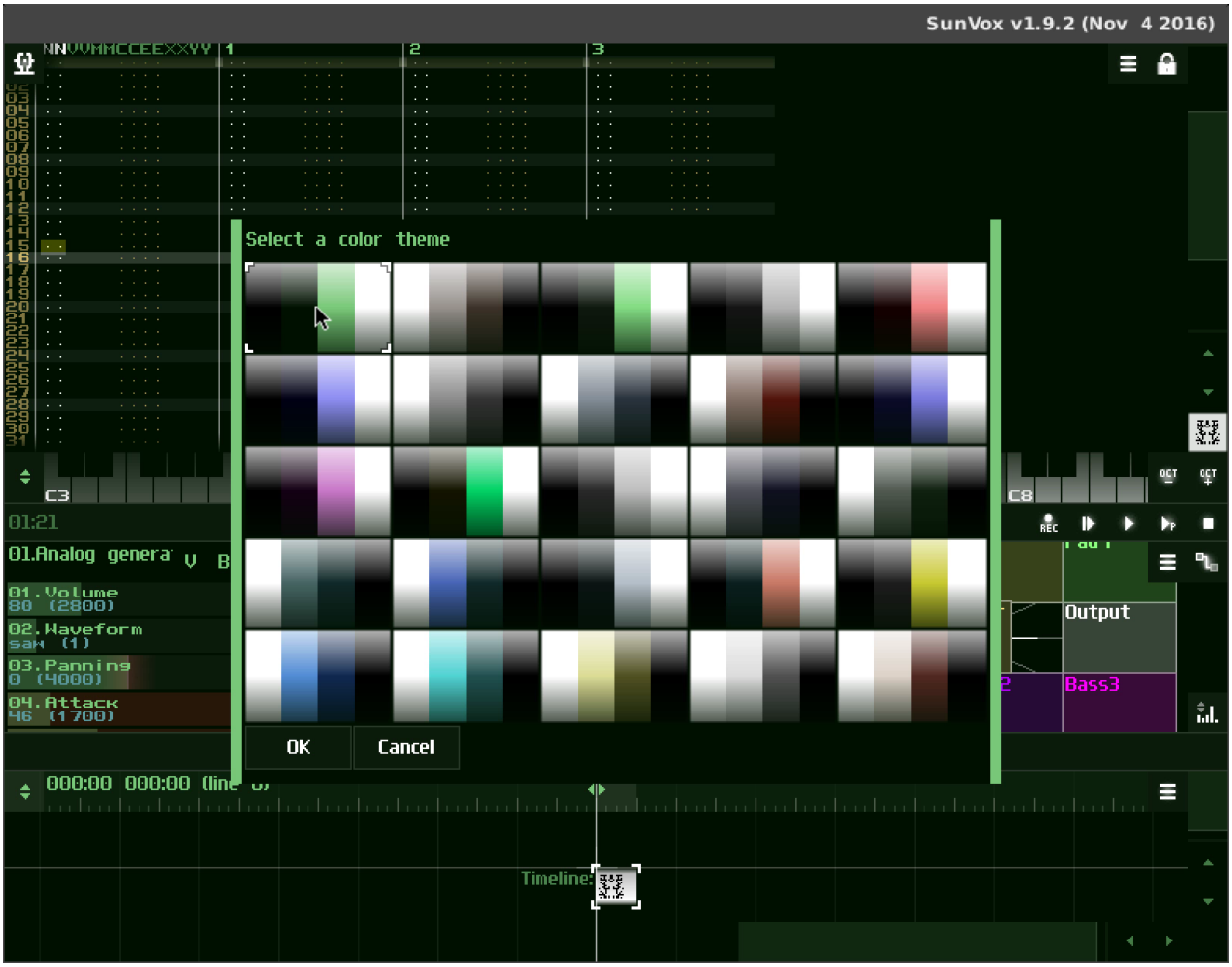

Fig. 3.1: SunVox color theme selection window.

If you choose a color scheme other than the default, you'll be asked to restart SunVox.

Fig. 3.2: Selecting a new color theme.

To choose a new color theme, use the main menu to select *Preferences*. Choose the *Interface* section, then click *Color theme*.

Fig. 3.3: Opening the color theme picker.

Contributors to this page

Author(s) Matthew Scott

3.3 Your first sounds [TBW]

3.3.1 Selecting a module

3.3.2 Playing notes using the on-screen keys

3.3.3 Playing notes with a keyboard

3.3.4 Playing notes with a MIDI device

3.4 Configuring MIDI inputs (optional) [TBW]

3.5 Terminology [TBW]

3.5.1 Project

3.5.2 Module

3.5.3 Controller

3.5.4 Connection

3.5.5 Timeline

3.5.6 Pattern

3.5.7 Row

3.5.8 Track

3.5.9 Note

3.5.10 Velocity

3.5.11 Effect

3.5.12 XX, YY, and XXYY values

3.6 Navigating the interface [TBW]

3.6.1 Using a mouse

3.6.2 Using multitouch

3.6.3 Using a keyboard

3.7 Creating an empty project [TBW]

3.8 Loading Modules [TBW]

3.9 Saving Projects [TBW]

CHAPTER 4

Beginner techniques [WIP]

4.1 Introduction to hex notation

Hex notation is just another way of writing numbers.

In normal decimal notation (what you're probably used to) you have:

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	...

In *hexadecimal* (or *hex*, for short) this becomes:

0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	10	11	12	13
14	15	16	17	18	19	1a	1b	1c	...

The way this works is that instead of running from 0 through 9, digits run from 0 through 15. However, since we don't have digits for 10, 11, 12, 13, 14 or 15, we use letters. So, a means 10, b means 11 and so on.

This change also extends to the positions of digits. In regular decimal notation, if you have two digits the first one counts tens. If you have three digits, the first one counts hundreds, the second counts tens and so on.

For example, if we have the number 14 in decimal, that can also be shown mathematically as:

$$1 \times 10 + 4 \times 1 = 14$$

In hexadecimal, you don't need a special position to count tens, because you can simply write "a" and have the same as 10 in decimal. Instead, you have a digit that counts by 16, or 16^1. The next digit up from that counts by 256, or 16^2. Next comes the digit that counts by 4096, or 16^3.

This is why 14 in hexadecimal corresponds to 20 in decimal:

$$1 \times 16 + 4 \times 1 = 20$$

When we rearrange our hex table above based on what we just learned, it looks like this:

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
20	21	22	23	24	25	26	27	28	29	2a	2b	2c	2d	2e	...

See also:

Hex conversion tables

4.1.1 Why do we use hex notation while tracking?

People use hex notation in tracking software for technical and historic reasons. This differs from many other types of DAW software, which often use decimal.

Electronic computers use the *binary* system of counting, to indicate on/off or +/-. The only two digits available are 0 and 1, which leads to long numbers like 11001 (which is 25 in decimal). Hexadecimal is a way to show those numbers very concisely, while keeping a mathematical relationship to the number 2.

The first tracking apps from the 1980s ran on computers far slower than today's, with limited screen space. Today, SunVox aims to support low-resolution screens and slower processors.

Hex gives you two advantages here:

1. You can show the full range of numbers of a "byte" using two digits (00-ff) instead of three with decimal (0-255).
2. You can work more directly with what's in the computer's memory, which can lead to smaller and more efficient code.

Interestingly, it also happens to fit well with the quarter note rhythms typical of music! After understanding, then practice, it will be familiar and comfortable for you to use.

4.1.2 How SunVox helps you use hex notation [TBW]

Contributors to this page

Author(s) Jan Koekepan, Matthew Scott

Editor(s) Matthew Scott

4.2 Adding and connecting modules [WIP]

Open sunvox, and look at your screen layout. You can see that it is divided into two, three or four panes, separated strips. You can tap and drag, or click and drag these strips to resize the panes. Make sure that none of the panes are hidden, by dragging the strips around so that you can see all the panes.

If you can see three panes, you may have your interface set not to show the timeline pane by default. On small screens, this is efficient but on larger screens it does not help you much. If you want to change this, go to the main menu -> preferences -> timeline, and change the Show Timeline setting.

If you see four panes, look at the middle two. The one one the right hand side, that contains small oblongs (at least one called Output) is where we will work.

If you see three panes, it's the one in the bottom right hand corner. If you see two panes, then the bottom one is the timeline, and is not what we need right now. You should see an icon at the top right hand corner of the timeline pane,

which looks like two boxes connected by a dogleg line. Touch or click that icon to get to the three pane view with the module pane.

Let's get a clean start. Open the main menu, and select new project. This should open a little box that lets you choose Template, Empty or Cancel. Select Empty.

You should now see a module pane that is blank, except for an Output module. We will create a basic module setup in this space now. Double-tap, or right-click in the pane, on a blank spot. You should have a menu.

If you used a mouse, you should have a menu that offers you the chance to add a module. Select this, and you should see a list of modules you can add.

If you used your finger, it should jump straight ahead to where you see a list of modules to add.

Select the Generator module, and hit OK. You should see a new Generator module in your view, marked with some blinking brackets at its corners.

Let's do that again, and this time add an Echo module.

We want a signal to go from the Generator, to the Echo, to the Output. This will give us a sound from the Generator, that echoes inside the Echo module, and will end up in Output so that we can hear it.

Touch or click on Generator so that you see flashing brackets at its corners. Then click or touch the module link button in the upper right hand corner of the pane (it looks like two boxes connected by a dogleg line), then click or touch the Echo module. You should see a line running from Generator to Echo, with little lights travelling in that direction.

Now do the same again, highlighting Echo and drawing a line to Output.

In the top pane, do you see the keyboard? It looks like a piano keyboard. Touch that, to hear your sound. Can you hear it echo? That means you did this correctly - this is the basic groundwork of adding modules to design your own sounds in sunvox.

Contributors to this page

Author(s) Jan Koekepan

4.3 Changing controller values [TBW]

Every module except for the Output module has some controllers. These controllers have a number of functions depending on the type of module. Some alter the synthesis of sound, some change effects - there are many purposes for these controllers, and if you want the details on each one, check the reference section of this book.

To the left of the module pane is the control pane. In here you can see the controls for whichever module you have selected. With a few exceptions, these controls appear as sliders. You can tap on them with a finger and drag them back and forth, or click and drag with a mouse.

If you want to, you can also right click on the controls or double tap, and open the Controller properties window, in which you can either move a slider for the controller value, or type one in directly.

Contributors to this page

Author(s) Jan Koekepan

4.4 Creating patterns in the timeline [WIP]

Make sure you have the timeline pane open at the bottom of your interface. If you are running sunvox on a small screen, you may have to switch from the module view to the pattern view.

On a touch interface, double-tap on the timeline where you want the new pattern. You should see it appear right where you double-tapped. It will have an automatically generated new icon on it, so that you can differentiate it from other patterns.

On a computer with a mouse, right-click on the background of the timeline, and when the menu opens, select New pattern.

In each case, you may want to tweak the pattern's appearance and length, so while your chosen pattern is highlighted, select the menu icon for the timeline (horizontal bars, near the top right-hand corner of the timeline pane) and select Pattern properties. On a computer with a mouse, you can also right-click to get the menu.

In the Pattern Properties menu, you can give the pattern a name (I like to make it something useful, like: Bassline A Major), change the pattern's length and the number of tracks, and alter the icon.

Contributors to this page

Author(s) Jan Koekepan

4.5 Editing patterns [TBW]

4.6 Adding controller changes to patterns [TBW]

You already know that you can change controllers on Sunvox modules.

The question is: can you write those changes into patterns so that the changes happen during playback of a song?

Yes, this is possible. Here is how:

The first option is to put sunvox into record mode, and while it records, just change the controllers that you want to change. Sunvox will record your changes into a new pattern.

The second option is to open the Controller properties window, set the value to what you want, and then use the Write to pattern button. The value that you selected will then be written as a command into the active pattern where your cursor was.

When you play back the song, that change in the controller value will take effect at that point in the song.

If you know the syntax, you can even type the value into the pattern directly, but you don't have to when the Controller properties window will do it for you.

Contributors to this page

Author(s) Jan Koekepan

4.7 Adding note effects [TBW]

4.8 Recording patterns from note input [TBW]

Sometimes you do not want to directly edit in every note. You know what you want to play, and how to get it in there, and you just want to record it into a pattern. Fortunately, this is easy in Sunvox.

At the bottom of your screen, make sure that you can see the timeline pane. Then click or tap to get the play head approximately where you want to start recording. I like to start a few seconds ahead. Also, make sure that you have selected the module for which you want to add more sound.

Then go up to the transport controls, under the virtual keyboard, and hit record. Sunvox will play back what you already have, and you can start to play new notes. Sunvox will create a new pattern that incorporates your additional notes.

Once you have finished, hit the stop button in the transport controls. You should see your new pattern in the timeline. Go back to play it, and you should hear your new notes. If you want to tweak what you did, simply edit the pattern directly as you would any other pattern.

Contributors to this page

Author(s) Jan Koekepan

4.9 Arranging patterns in the timeline [WIP]

The timeline is where you arrange your music. As the music plays, you can watch the playhead bar moving along the timeline. If you want to get your sounds happening at the right time, you need to move the patterns that play those sounds to the right place in the timeline.

You can use a mouse to click and drag a pattern around the timeline, or your finger to touch and drag them the same way.

Vertical positioning of a pattern in the timeline does not matter. This means that you can put a few patterns next to each other, vertically arranged, and the music in them will play at the same time.

Horizontal positioning of the pattern affects when in your music the sounds in that pattern will play. This is also why longer patterns are horizontally longer in the pattern display.

Contributors to this page

Author(s) Jan Koekepan

4.10 Changing pattern properties [TBW]

4.11 Exporting to WAV files [WIP]

Sunvox can render your music to a .WAV file so that you can copy it, add it to other mixes, transcode it or whatever else you want to do.

Select the main menu in the upper left hand corner, then choose Export/import. This should open a little box with two options:

- Export to WAV
- Export to MIDI

What we want here is to export to WAV, so select that one.

This opens another window called Select export format.

You can choose your bit depth. There are two options: 16 bit (integer) and 32 bit (floating point). If you don't know which to choose, 16 bit is not going to get you in any trouble. That's CD quality. 32 bit is overkill for most purposes.

You can also choose your export mode:

- One file (Output)
- One file (selected module)
- File per module
- File per module (effects)
- File per module (connected to Output)

If you just want to hear your song, choose One file (Output).

Next click the Export button. This will open a window in which you can name your file, and sunvox will render your file for you. Once this is done, you can copy it, listen to it - whatever you like!

The other export modes meet other needs, and are a little more advanced.

One file (selected module) lets you just get the sound that came from one module in your song, rather than just all the sound that emerged from Output.

File per module is kind of like selected module, except that it will generate a separate WAV file for every single module. Careful, on a big project this can take a lot of storage! Each filename will start with a name that you gave, then the module's number, then the module type, for example:

    MySong_01_Generator.wav

File per module (effects) is like file per module, but only creates the output of each effects module. This can make sense if you're only interested in the post-effects output of every sound, rather than saving the raw sounds as well.

File per module (connected to Output) lets you save the output of every module that is linked to the Output module. This can make a lot of sense if you want the final sounds that reached the Output module, but you want them broken out.

In general, the other options are for musicians who want to extract stems from a sunvox song so that they can apply another stage of mastering. If you use sunvox as your end-to-end musical solution, you probably won't need these. Just save one file, from Output, and show the world your music.

Contributors to this page

Author(s) Jan Koekepan

CHAPTER 5

Intermediate techniques [TBW]

5.1 Customizing the starting template [TBW]

5.2 Combining note effects [TBW}

5.3 File management [TBW]

5.4 Pattern management [TBW]

5.5 Clipboards [TBW]

CHAPTER 6

Advanced techniques [TBW]

6.1 Mastering [TBW]

6.2 Live Performance [TBW]

6.3 Tuning and Microtonal Music [TBW]

6.4 Building MetaModules [TBW]

CHAPTER 7

SunVox Cookbook

This is a collection of recipes and tutorials that didn't fit elsewhere in this book.

7.1 Adding swing to your project

Swing, also known as shuffle or groove, refers to stretching and shortening alternating notes of a rhythm. It's often used to add a more relaxed feel to a song.

This recipe will focus on adding swing to 16th notes, using the default 4/4 time and 16th note grid in SunVox. We'll cover two ways to add swing, and list the benefits and drawbacks of each method.

7.1.1 Common principles

PPQN: Parts per quarter note

By default, lines in SunVox represent 16th notes. Also by default, each line is divided into six ticks. This means each quarter note is divided into 24 ticks, or *24 PPQN (parts per quarter note)*.

24 PPQN is the default in SunVox for historical and practical reasons. However, this gives you a limited range of swing amounts.

For adding more subtle swing to your project, 96 PPQN is useful. Thankfully, SunVox supports this by changing the BPM and TPL in project properties.

One way of doing this, while keeping 16th note lines in your patterns, is to multiply both BPM and TPL in your project properties by 4.

The example project used here has an actual BPM of 80, so the project's BPM is set to 320 and the TPL is set to 12.

Swing percentages

Swing is often referred to as a percentage between 50% and 75%. If an 8th note of time in a track represents 100%, the swing percentage is the amount of time the first 16th note will occupy. The second 16th note is then triggered and occupies the remainder.

Without adding any swing, your swing is at *50%*. This means that both 16th notes are the same length:

```
          1111111111222222222233333333344444444
 012345678901234567890123456789012345678901234567   <-- tick (96 PPQN)
+----------------------><------------------------+
|<-50%----------------><-50%------------------>|  <-- 8th note divided into
|<-24 ticks-----------><-24 ticks------------->|      equal 16th notes
+----------------------><------------------------+
 \__________/\__________/\__________/\__________/   <-- 32nd notes
 0.......................1.......................   <-- line (16th note)
```

If we have a swing of *75%*, what effectively happens is that the first 16th note becomes dotted (which means it takes up 1.5 times the space of a 16th note) and the second becomes a 32nd note:

```
          1111111111222222222233333333344444444
 012345678901234567890123456789012345678901234567
+-----------------------------------><-----------+
|<-75%--------------------------------><-25%------>|
|<-36 ticks-------------------------><-12 ticks->|
+-----------------------------------><-----------+
 \__________/\__________/\__________/\__________/
 0.......................1.......................
```

This creates a very extreme swing, so typically something in between is used, such as 62.5%, sometimes shown as "62%" on hardware sequencers:

```
          1111111111222222222233333333344444444
 012345678901234567890123456789012345678901234567
+-----------------------------><-----------------+
|<-62.5%----------------------><-37.5%---------->|
|<-30 ticks-------------------><-18 ticks------->|
+-----------------------------><-----------------+
 \__________/\__________/\__________/\__________/
 0.......................1.......................
```

Now that you have knowledge of how swing works musically, you can explore these methods for adding swing to your SunVox projects.

7.1.2 Example project without swing

As a point of comparison, here is what our example project sounds like before adding any swing. In other words, its swing value is 50%.

`Download Audio for No Swing Applied`

`Download No Swing Applied`

Modules

We have a basic setup involving a FM synth passed through a couple of filters, one of which has a line-based LFO running. We also have a drum synth that's providing a kick, snare, and hihat.

FM pattern

```
   | NN VV MM CC EE XXYY
00 | G4    02
01 | B3    02
02 | a3    02
03 | //
04 | a3    02
05 | //
06 | a3    02
07 | //
08 | a3    02
09 | //
10 | a3    02
11 | //
12 | a3    02
13 | a3    02
14 | a3    02
15 | //
16 | G4    02
17 | B3    02
18 | a3    02
19 | //
20 | a3    02
21 | //
22 | a3    02
23 | //
24 | G3    02
25 | //
26 | a3    02
27 | //
28 | a3    02
29 | G3    02
30 | f3    02
31 | //
32 | G4    02
33 | B3    02
34 | a3    02
35 | //
36 | a3    02
37 | //
38 | a3    02
39 | //
40 | a3    02
41 | //
42 | a3    02
43 | //
44 | a3    02
45 | a3    02
46 | a3    02
47 | //
48 | G4    02
49 | B3    02
50 | a3    02
51 | //
```

```
52 | a3    02
53 | //
54 | a3    02
55 | //
56 | G3    02
57 | //
58 | a3    02
59 | //
60 | a3    02
61 | G3    02
62 | f3    02
63 | //
```

LFO initialization pattern

```
   | NN VV MM CC EE XXYY
00 | ..    03 10 00
01 | ..
02 | ..
03 | ..
04 | ..
05 | ..
06 | ..
07 | ..
08 | ..
09 | ..
10 | ..
11 | ..
12 | ..
13 | ..
14 | ..
15 | ..
16 | ..
17 | ..
18 | ..
19 | ..
20 | ..
21 | ..
22 | ..
23 | ..
24 | ..
25 | ..
26 | ..
27 | ..
28 | ..
29 | ..
30 | ..
31 | ..
```

DrumSynth pattern

```
   | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY
00 | D2    01            | f2    01            | G6    01
01 | //                  | f2    01            | //
```

```
02 | //                  | E5    01            | //
03 | //                  | f2    01            | //
04 | D2    01            | f2    01            | G6    01
05 | //                  | f2    01            | //
06 | //                  | E5    01            | //
07 | //                  | f2    01            | //
08 | D2    01            | f2    01            | G6    01
09 | //                  | f2    01            | //
10 | //                  | E5    01            | //
11 | //                  | f2    01            | //
12 | D2    01            | f2    01            | G6    01
13 | //                  | f2    01            | G6    01
14 | //                  | f2    01            | G6    01
15 | //                  | E5    01            | //
16 | D2    01            | f2    01            | G6    01
17 | //                  | f2    01            | //
18 | //                  | E5    01            | //
19 | //                  | f2    01            | //
20 | D2    01            | f2    01            | G6    01
21 | //                  | f2    01            | //
22 | //                  | E5    01            | //
23 | //                  | f2    01            | //
24 | D2    01            | f2    01            | G6    01
25 | //                  | f2    01            | //
26 | //                  | E5    01            | //
27 | //                  | f2    01            | //
28 | D2    01            | f2    01            | G6    01
29 | //                  | f2    01            | G5    01
30 | //                  | f2    01            | //
31 | D2    01            | E5    01            | //
32 | D2    01            | f2    01            | G6    01
33 | //                  | f2    01            | //
34 | //                  | E5    01            | //
35 | //                  | f2    01            | //
36 | D2    01            | f2    01            | G6    01
37 | //                  | f2    01            | //
38 | //                  | E5    01            | //
39 | //                  | f2    01            | //
40 | D2    01            | f2    01            | G6    01
41 | //                  | f2    01            | //
42 | //                  | E5    01            | //
43 | //                  | f2    01            | //
44 | D2    01            | f2    01            | G6    01
45 | //                  | f2    01            | G6    01
46 | //                  | f2    01            | G6    01
47 | //                  | E5    01            | //
48 | D2    01            | f2    01            | G6    01
49 | //                  | f2    01            | //
50 | //                  | E5    01            | //
51 | //                  | f2    01            | //
52 | D2    01            | f2    01            | G6    01
53 | //                  | f2    01            | //
54 | //                  | E5    01            | //
55 | //                  | f2    01            | //
56 | D2    01            | E5    01            | G6    01
57 | //                  | f2    01            | //
58 | //                  | f2    01            | //
59 | //                  | E5    01            | //
```

```
60 | D2    01             | f2    01             | G5    01
61 | D2    01             | f2    01             | //
62 | D2    01             | E5    01             | //
63 | //                   | f2    01             | //
```

7.1.3 Method 1: Global swing by setting TPL

This method works by alternating the number of ticks each line takes. The first line sets it to the longer value, and the second line sets it to the shorter.

It will affect all notes in all patterns globally, and you will also see the swing visually during playback.

Applying the method

Find the "1st note ticks" and "2nd note ticks" that match your PPQN and desired swing amount. (See *Tables*)

Create a new pattern that alternates between the two values using the *0F: Set playing speed [TBW]* note effect.

Clone the pattern as many times as needed to reach the end of your song (or the end of the section where you want swing).

Here's what it looks like to apply 56% swing:

```
   | NN VV MM CC EE XXYY
00 | ..       00 0F 001B
01 | ..       00 0F 0015
02 | ..       00 0F 001B
03 | ..       00 0F 0015
04 | ..       00 0F 001B
05 | ..       00 0F 0015
06 | ..       00 0F 001B
07 | ..       00 0F 0015
08 | ..       00 0F 001B
09 | ..       00 0F 0015
10 | ..       00 0F 001B
11 | ..       00 0F 0015
12 | ..       00 0F 001B
13 | ..       00 0F 0015
14 | ..       00 0F 001B
15 | ..       00 0F 0015
16 | ..       00 0F 001B
17 | ..       00 0F 0015
18 | ..       00 0F 001B
19 | ..       00 0F 0015
20 | ..       00 0F 001B
21 | ..       00 0F 0015
22 | ..       00 0F 001B
23 | ..       00 0F 0015
24 | ..       00 0F 001B
25 | ..       00 0F 0015
26 | ..       00 0F 001B
27 | ..       00 0F 0015
28 | ..       00 0F 001B
29 | ..       00 0F 0015
30 | ..       00 0F 001B
31 | ..       00 0F 0015
```

Here's what it sounds like:

Download Audio for Method 1, 96 PPQN, 56% swing

Download Method 1, 96 PPQN, 56% swing

Notice how the filter's LFO no longer aligns the same way as before. This is one of the side-effects of this method and is discussed in the list of drawbacks below.

Advantages

- Requires only a single track to apply swing to all notes.
- Swing across entire project can be adjusted quickly.
- All note effects may be used.

Drawbacks

- All tracks must follow the same swing amount. Track-specific swing, as described below, can allow for this. Mixing the two techniques is left as an exercise for the reader.
- Maximum swing value is 64% when using 96 PPQN, as TPL cannot be set past 31.
- If you have any LFOs set to use "line", "line/2", or "line/3" as a frequency unit, they will be affected by the shifting TPL. You may hear audible clicks as a result. Use a different frequency unit, or try track-specific swing instead. Affected modules: *Delay [TBW]*, *Echo [TBW]*, *Filter [TBW]*, *Filter Pro [TBW]*, *Flanger [TBW]*, *LFO [TBW]*, *Vibrato [TBW]*.

7.1.4 Method 2: Track-specific swing by delaying notes

This method works by alternating the number of ticks each line takes. The first line sets it to the longer value, and the second line sets it to the shorter.

It will affect all notes in all patterns globally, and you will also see the swing visually during playback.

Applying the method

Find the "2nd note delay" that matches your PPQN and desired swing amount. (See *Tables*)

Delay every second line in the sequence of notes you want to swing using the *1D: Delay start during line [TBW]* note effect.

Here's what it looks like to apply 56% swing to our FM bassline, 52% to our kick drum, 62% to our hihats, and 75% to our snare:

```
   | NN VV MM CC EE XXYY
00 | g4    02
01 | B3    02 00 1D 0003
02 | a3    02
03 | //       00 1D 0003
04 | a3    02
05 | //       00 1D 0003
06 | a3    02
07 | //       00 1D 0003
08 | a3    02
09 | //       00 1D 0003
```

```
10 | a3    02
11 | //       00 1D 0003
12 | a3    02
13 | a3    02 00 1D 0003
14 | a3    02
15 | //       00 1D 0003
16 | G4    02
17 | B3    02 00 1D 0003
18 | a3    02
19 | //       00 1D 0003
20 | a3    02
21 | //       00 1D 0003
22 | a3    02
23 | //       00 1D 0003
24 | G3    02
25 | //       00 1D 0003
26 | a3    02
27 | //       00 1D 0003
28 | a3    02
29 | G3    02 00 1D 0003
30 | f3    02
31 | //       00 1D 0003
32 | G4    02
33 | B3    02 00 1D 0003
34 | a3    02
35 | //       00 1D 0003
36 | a3    02
37 | //       00 1D 0003
38 | a3    02
39 | //       00 1D 0003
40 | a3    02
41 | //       00 1D 0003
42 | a3    02
43 | //       00 1D 0003
44 | a3    02
45 | a3    02 00 1D 0003
46 | a3    02
47 | //       00 1D 0003
48 | G4    02
49 | B3    02 00 1D 0003
50 | a3    02
51 | //       00 1D 0003
52 | a3    02
53 | //       00 1D 0003
54 | a3    02
55 | //       00 1D 0003
56 | G3    02
57 | //       00 1D 0003
58 | a3    02
59 | //       00 1D 0003
60 | a3    02
61 | G3    02 00 1D 0003
62 | f3    02
63 | //       00 1D 0003
```

```
   | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY
00 | D2    01            | f2    01            | G6    01
01 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
```

```
02 | //                  | E5    01            | //
03 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
04 | D2    01            | f2    01            | G6    01
05 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
06 | //                  | E5    01            | //
07 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
08 | D2    01            | f2    01            | G6    01
09 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
10 | //                  | E5    01            | //
11 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
12 | D2    01            | f2    01            | G6    01
13 | //       00 1D 0001 | f2    01 00 1D 0006 | G6    01 00 1D 000C
14 | //                  | f2    01            | G6    01
15 | //       00 1D 0001 | E5    01 00 1D 0006 | //       00 1D 000C
16 | D2    01            | f2    01            | G6    01
17 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
18 | //                  | E5    01            | //
19 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
20 | D2    01            | f2    01            | G6    01
21 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
22 | //                  | E5    01            | //
23 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
24 | D2    01            | f2    01            | G6    01
25 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
26 | //                  | E5    01            | //
27 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
28 | D2    01            | f2    01            | G6    01
29 | //       00 1D 0001 | f2    01 00 1D 0006 | G5    01 00 1D 000C
30 | //                  | f2    01            | //
31 | D2    01 00 1D 0001 | E5    01 00 1D 0006 | //       00 1D 000C
32 | D2    01            | f2    01            | G6    01
33 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
34 | //                  | E5    01            | //
35 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
36 | D2    01            | f2    01            | G6    01
37 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
38 | //                  | E5    01            | //
39 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
40 | D2    01            | f2    01            | G6    01
41 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
42 | //                  | E5    01            | //
43 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
44 | D2    01            | f2    01            | G6    01
45 | //       00 1D 0001 | f2    01 00 1D 0006 | G6    01 00 1D 000C
46 | //                  | f2    01            | G6    01
47 | //       00 1D 0001 | E5    01 00 1D 0006 | //       00 1D 000C
48 | D2    01            | f2    01            | G6    01
49 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
50 | //                  | E5    01            | //
51 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
52 | D2    01            | f2    01            | G6    01
53 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
54 | //                  | E5    01            | //
55 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
56 | D2    01            | E5    01            | G6    01
57 | //       00 1D 0001 | f2    01 00 1D 0006 | //       00 1D 000C
58 | //                  | f2    01            | //
59 | //       00 1D 0001 | E5    01 00 1D 0006 | //       00 1D 000C
```

```
60 | D2    01             | f2    01             | G5    01
61 | D2    01 00 1D 0001  | f2    01 00 1D 0006  | //       00 1D 000C
62 | D2    01             | E5    01             | //
63 | //       00 1D 0001  | f2    01 00 1D 0006  | //       00 1D 000C
```

Here's what it sounds like:

`Download Audio for Method 2, 96 PPQN, variable swing`

`Download Method 2, 96 PPQN, variable swing`

This example may be a bit overdone, but it shows how you can use different swing values simultaneously.

The filter's LFO aligns the same way as without swing, because we are not changing the size of lines.

Advantages

- Simpler way to have different swing amounts for different tracks.
- LFOs using "line", "line/2", or "line/3" as a frquency unit will not be affected as the TPL will stay fixed.

Drawbacks

- Must place note effects in an adjacent track.
- More difficult to change swing values once set.
- Some note effects may not be used: 20-23 (random notes and controller values), 40-5f (delay event for line fraction).

7.1.5 Tables

Here is a table of swing percentage values. It'll be useful for the techniques explained below. Each value shows the number of ticks each 16th note will take, and the number of ticks the second 16th note will need to be delayed.

If you're using a TPL other than 6, 12, or 24, I applaud you for your creativity! You'll have to calculate swing and BPM on your own, but the same methods described above will still apply.

24 PPQN

Keep TPL at 6.

%	1st note ticks	2nd note ticks	2nd note delay
50	6	6	0
58	7	5	1
66	8	4	2
75	9	3	3

48 PPQN

Set TPL to 12, multiply project BPM by 2.

%	1st note ticks	2nd note ticks	2nd note delay
50	12 (c)	12 (c)	0 (0)
54	13 (d)	11 (b)	2 (2)
58	14 (e)	10 (a)	4 (4)
62	15 (f)	9 (9)	6 (6)
66	16 (10)	8 (8)	8 (8)
70	17 (11)	7 (7)	10 (a)
75	18 (12)	6 (6)	12 (c)

96 PPQN

Set TPL to 24, multiply project BPM by 4.

Note: Maximum TPL is 31, so 64% swing is highest available using *Method 1: Global swing by setting TPL*.

%	1st note ticks	2nd note ticks	2nd note delay
50	24 (18)	24 (18)	0 (0)
52	25 (19)	23 (17)	1 (1)
54	26 (1a)	22 (16)	2 (2)
56	27 (1b)	21 (15)	3 (3)
58	28 (1c)	20 (14)	4 (4)
60	29 (1d)	19 (13)	5 (5)
62	30 (1e)	18 (12)	6 (6)
64	31 (1f)	17 (11)	7 (7)
66			8 (8)
68			9 (9)
70			10 (a)
72			11 (b)
75			12 (c)

7.1.6 Further reading

For more about the history behind swing and quantization, read this interview with Roger Linn, creator of the LinnDrum drum machine and designer of early Akai MPC workstations.

Contributors to this page

Author(s) Matthew Scott

7.2 Reverse Percussion

Reversing percussion sounds can add uniqueness to a driving beat, such as around phrase transitions, or to add an element of surprise in the middle of a phrase.

7.2.1 Kick Drums

One particular style of reversing kicks sounds kind of like a "whip", as demonstrated by the very end of this phrase transition in SP23's "Leave This City Kickin":

`Download Reverse Kick Demonstration`

This recipe shows you a way you can take a synth module, then use the ping-pong setting of a loop module to create an isolated reversal.

`Download Audio for Reverse Kick Project`

`Download Reverse Kick Project`

Modules

kicker Provides a constant kick drum rhythm.

kick amp Turned on when we want to hear the forward kick drum, off when we want to reverse.

kick loop Provides a constant ping-pong loop of the kicker.

kick loop amp Turned off normally, and momentarily turned on to reveal the reverse portion of the ping-pong loop.

drumsynth Hi-hat and snare to fill out the rhythm.

Patterns

kick

Provides a constant kick drum rhythm via the kicker module. This pattern is repeated throughout the example.

kick on and *kick off* control the presence of the kicker via the kick amp module.

```
   | NN VV MM CC EE XXYY
00 | G2    01
01 | //
02 | //
03 | //
04 | G2    01
05 | //
06 | //
07 | //
08 | G2    01
09 | //
10 | //
11 | //
12 | G2    01
13 | //
14 | //
15 | //
16 | G2    01
17 | //
18 | //
19 | //
20 | G2    01
21 | //
```

```
22 | //
23 | //
24 | G2    01
25 | //
26 | //
27 | //
28 | G2    01
29 | //
30 | //
31 | //
```

perc

Hi-hat and snare to fill out the rhythm.

```
   | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY
00 | f4    02            | ==                  | ..
01 | f4    02            | ..                  | ..
02 | //                  | F4    02            | ..
03 | //                  | //                  | ..
04 | f4    02            | ==                  | G5    02
05 | f4    02            | ..                  | //
06 | //                  | F4    02            | //
07 | //                  | //                  | //
08 | f4    02            | ==                  | //
09 | f4    02            | ..                  | //
10 | //                  | F4    02            | //
11 | //                  | //                  | //
12 | f4    02            | ==                  | G5    02
13 | f4    02            | ..                  | //
14 | //                  | F4    02            | //
15 | //                  | //                  | //
16 | f4    02            | ==                  | //
17 | f4    02            | ..                  | //
18 | //                  | F4    02            | //
19 | //                  | //                  | //
20 | f4    02            | ==                  | G5    02
21 | f4    02            | ..                  | //
22 | //                  | F4    02            | //
23 | //                  | //                  | //
24 | f4    02            | ==                  | //
25 | f4    02            | ..                  | //
26 | //                  | F4    02            | //
27 | //                  | //                  | //
28 | f4    02            | ==                  | G5    02
29 | f4    02            | ..                  | //
30 | //                  | F4    02            | //
31 | //                  | //                  | //
```

perc halt

A variation of *perc* to give room for the reverse kick.

```
   | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY
00 | f4    02            | ==                  | ..
01 | f4    02            | ..                  | ..
02 | //                  | F4    02            | ..
03 | //                  | //                  | ..
04 | f4    02            | ==                  | G5    02
05 | f4    02            | ..                  | //
06 | //                  | F4    02            | //
07 | //                  | //                  | //
08 | f4    02            | ==                  | //
09 | f4    02            | ..                  | //
10 | //                  | F4    02            | //
11 | //                  | //                  | //
12 | f4    02            | ==                  | G5    02
13 | f4    02            | ..                  | //
14 | //                  | F4    02            | //
15 | //                  | //                  | //
16 | f4    02            | ==                  | //
17 | f4    02            | ..                  | //
18 | //                  | F4    02            | //
19 | //                  | //                  | //
20 | f4    02            | ==                  | G5    02
21 | f4    02            | ..                  | //
22 | //                  | F4    02            | //
23 | //                  | //                  | //
24 | f4    02            | ==                  | //
25 | f4    02            | ..                  | //
26 | //                  | F4    02            | //
27 | //                  | //                  | //
28 | //                  | //                  | G5    02
29 | //                  | //                  | //
30 | //                  | //                  | //
31 | //                  | //                  | //
```

kick on

Turns on the kick amp, and resets the envelope acceleration of the kicker.

```
   | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY
00 | ..    03 01 00 2000 | ..    05 01 00      | ..    01 05 00 2000
01 | ..                  | ..                  | ..
02 | ..                  | ..                  | ..
03 | ..                  | ..                  | ..
04 | ..                  | ..                  | ..
05 | ..                  | ..                  | ..
06 | ..                  | ..                  | ..
07 | ..                  | ..                  | ..
08 | ..                  | ..                  | ..
09 | ..                  | ..                  | ..
10 | ..                  | ..                  | ..
11 | ..                  | ..                  | ..
12 | ..                  | ..                  | ..
13 | ..                  | ..                  | ..
14 | ..                  | ..                  | ..
15 | ..                  | ..                  | ..
```

kick off

Turns off the kick amp.

```
   | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY
00 | ..    03 01 00      | ..    05 01 00
01 | ..                  | ..
02 | ..                  | ..
03 | ..                  | ..
04 | ..                  | ..
05 | ..                  | ..
06 | ..                  | ..
07 | ..                  | ..
```

kick loop

Controls the reverse kick loop, and modifies the envelope acceleration of the kicker so that the reversed kick sounds a little different than the forward kick that plays immediately after.

```
   | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY | NN VV MM CC EE XXYY
00 | ..                  | ..                  | ..
01 | ..                  | ..                  | ..
02 | ..                  | ..                  | ..
03 | ..                  | ..                  | ..
04 | ..                  | ..                  | ..
05 | ..                  | ..                  | ..
06 | ..                  | ..                  | ..
07 | ..                  | ..                  | ..
08 | ..                  | ..                  | ..
09 | ..                  | ..                  | ..
10 | ..                  | ..                  | ..
11 | ..                  | ..                  | ..
12 | ..    01 05 00 1000 | ..    04 04 00 0001 | ..    05 01 00
13 | ..                  | ..                  | ..
14 | ..                  | ..                  | ..    05 01 00 2000
15 | ..                  | ..                  | ..
```

Contributors to this page

Author(s) Matthew Scott

CHAPTER 8

Module reference [TBW]

8.1 Synths

8.1.1 Analog Generator [TBW]

Controller reference

class rv.modules.analoggenerator.**AnalogGenerator**(***kwargs*)

"Analog generator" SunVox Synth Module

Behaviors:

- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..256>	80
02 (2)	waveform	<enum 'Waveform'>	<Waveform.triangle: 0>
03 (3)	panning	<Range -128..128>	0
04 (4)	attack	<Range 0..256>	0
05 (5)	release	<Range 0..256>	0
06 (6)	sustain	<class 'bool'>	True
07 (7)	exponential_envelope	<class 'bool'>	True
08 (8)	duty_cycle	<Range 0..1024>	512
09 (9)	freq2	<Range 0..2000>	1000
0a (10)	filter	<enum 'Filter'>	<Filter.off: 0>
0b (11)	f_freq_hz	<Range 0..14000>	14000
0c (12)	f_resonance	<Range 0..1530>	0
0d (13)	f_exponential_freq	<class 'bool'>	True
0e (14)	f_attack	<Range 0..256>	0
0f (15)	f_release	<Range 0..256>	0
10 (16)	f_envelope	<enum 'FilterEnvelope'>	<FilterEnvelope.off: 0>
11 (17)	polyphony_ch	<Range 1..32>	16
12 (18)	mode	<enum 'Mode'>	<Mode.hq: 0>
13 (19)	noise	<Range 0..256>	0

class AnalogGenerator.**Waveform**

An enumeration.

Name	Value
triangle	0
saw	1
square	2
noise	3
drawn	4
sin	5
hsin	6
asin	7
drawn_with_spline_interpolation	8

class AnalogGenerator.**Filter**

An enumeration.

Name	Value
off	0
lp_12db	1
hp_12db	2
bp_12db	3
br_12db	4
lp_24db	5
hp_24db	6
bp_24db	7
br_24db	8

class AnalogGenerator.**FilterEnvelope**

An enumeration.

Name	Value
off	0
sustain_off	1
sustain_on	2

class `AnalogGenerator.`**`Mode`**
An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3

8.1.2 DrumSynth [TBW]

Controller reference

class `rv.modules.drumsynth.`**`DrumSynth`**`(`*`**kw`*`)`
"DrumSynth" SunVox Synth Module

Behaviors:

- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..512>	256
`02` (2)	panning	<Range -128..128>	0
`03` (3)	polyphony_ch	<Range 1..8>	4
`04` (4)	bass_volume	<Range 0..512>	200
`05` (5)	bass_power	<Range 0..256>	256
`06` (6)	bass_tone	<Range 0..256>	64
`07` (7)	bass_length	<Range 0..256>	64
`08` (8)	hihat_volume	<Range 0..512>	256
`09` (9)	hihat_length	<Range 0..256>	64
`0a` (10)	snare_volume	<Range 0..512>	256
`0b` (11)	snare_tone	<Range 0..256>	128
`0c` (12)	snare_length	<Range 0..256>	64

8.1.3 FM [TBW]

Controller reference

class `rv.modules.fm.`**`Fm`**`(`*`**kw`*`)`
"FM" SunVox Synth Module

Behaviors:

- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	c_volume	<Range 0..256>	128
02 (2)	m_volume	<Range 0..256>	48
03 (3)	panning	<Range -128..128>	0
04 (4)	c_freq_ratio	<Range 0..16>	1
05 (5)	m_freq_ratio	<Range 0..16>	1
06 (6)	m_feedback	<Range 0..256>	0
07 (7)	c_attack	<Range 0..512>	32
08 (8)	c_decay	<Range 0..512>	32
09 (9)	c_sustain	<Range 0..256>	128
0a (10)	c_release	<Range 0..512>	64
0b (11)	m_attack	<Range 0..512>	32
0c (12)	m_decay	<Range 0..512>	32
0d (13)	m_sustain	<Range 0..256>	128
0e (14)	m_release	<Range 0..512>	64
0f (15)	m_scaling_per_key	<Range 0..4>	0
10 (16)	polyphony_ch	<Range 1..16>	4
11 (17)	mode	<enum 'Mode'>	<Mode.hq: 0>

class Fm.**Mode**

An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3

8.1.4 Generator [TBW]

Controller reference

class rv.modules.generator.**Generator**(***kwargs*)

"Generator" SunVox Synth Module

Behaviors:

- receives_notes
- receives_modulator
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..256>	128
02 (2)	waveform	<enum 'Waveform'>	<Waveform.triangle: 0>
03 (3)	panning	<Range -128..128>	0
04 (4)	attack	<Range 0..512>	0
05 (5)	release	<Range 0..512>	0
06 (6)	polyphony_ch	<Range 1..16>	8
07 (7)	mode	<enum 'Mode'>	<Mode.stereo: 0>
08 (8)	sustain	<class 'bool'>	True
09 (9)	freq_modulation_input	<Range 0..256>	0
0a (10)	duty_cycle	<Range 0..1022>	511

class `Generator.`**`Waveform`**
An enumeration.

Name	Value
triangle	0
saw	1
square	2
noise	3
dirty	4
sin	5
hsin	6
asin	7
psin	8

class `Generator.`**`Mode`**
An enumeration.

Name	Value
stereo	0
mono	1

8.1.5 Input [TBW]

Controller reference

class `rv.modules.input.`**`Input`**`(`***kw*`)`
"Input" SunVox Synth Module

Behaviors:

- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..1024>	256
`02` (2)	channels	<enum 'Channels'>	<Channels.mono: 0>

class `Input.`**`Channels`**
An enumeration.

Name	Value
mono	0
stereo	1

8.1.6 Kicker [TBW]

Controller reference

class `rv.modules.kicker.`**`Kicker`**`(`***kw*`)`
"Kicker" SunVox Synth Module

Behaviors:

- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..256>	256
02 (2)	waveform	<enum 'Waveform'>	<Waveform.triangle: 0>
03 (3)	panning	<Range -128..128>	0
04 (4)	attack	<Range 0..512>	0
05 (5)	release	<Range 0..512>	32
06 (6)	vol_addition	<Range 0..1024>	0
07 (7)	env_acceleration	<Range 0..1024>	256
08 (8)	polyphony_ch	<Range 1..4>	1
09 (9)	anticlick	<class 'bool'>	False

class Kicker.**Waveform**
An enumeration.

Name	Value
triangle	0
square	1
sin	2

8.1.7 Sampler [TBW]

Controller reference

class rv.modules.sampler.**Sampler**(***kwargs*)
"Sampler" SunVox Synth Module

Note: Radiant Voices only supports sampler modules in files that were saved using newer versions of SunVox.

Files created using older versions of SunVox, such as some of the files in the `simple_examples` included with SunVox, must first be loaded into the latest version of SunVox and then saved.

Behaviors:

- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..512>	256
02 (2)	panning	<Range -128..128>	0
03 (3)	sample_interpolation	<enum 'SampleInterpolation'>	<SampleInterpolation.spline: 2>
04 (4)	envelope_interpolation	<enum 'EnvelopeInterpolation'>	<EnvelopeInterpolation.linear: 1>
05 (5)	polyphony_ch	<Range 1..32>	8
06 (6)	rec_threshold	<Range 0..10000>	4
07 (7)	vibrato_type	<enum 'VibratoType'>	<VibratoType.sin: 0>
08 (8)	vibrato_attack	<Range 0..255>	0
09 (9)	vibrato_depth	<Range 0..255>	0
0a (10)	vibrato_rate	<Range 0..63>	0
0b (11)	volume_fadeout	<Range 0..8192>	0

class Sampler.**SampleInterpolation**
An enumeration.

Name	Value
off	0
linear	1
spline	2

class Sampler.**EnvelopeInterpolation**
An enumeration.

Name	Value
off	0
linear	1

8.1.8 SpectraVoice [TBW]

Controller reference

class rv.modules.spectravoice.**SpectraVoice**(***kwargs*)
"SpectraVoice" SunVox Synth Module

Behaviors:

- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..256>	128
02 (2)	panning	<Range -128..128>	0
03 (3)	attack	<Range 0..512>	10
04 (4)	release	<Range 0..512>	512
05 (5)	polyphony_ch	<Range 1..32>	8
06 (6)	mode	<enum 'Mode'>	<Mode.hq_spline: 4>
07 (7)	sustain	<class 'bool'>	True
08 (8)	spectrum_resolution	<Range 0..5>	1
09 (9)	harmonic	<Range 0..15>	0
0a (10)	h_freq_hz	<Range 0..22050>	1098
0b (11)	h_volume	<Range 0..255>	255
0c (12)	h_width	<Range 0..255>	3
0d (13)	h_type	<enum 'HarmonicType'>	<HarmonicType.hsin: 0>

class SpectraVoice.**Mode**
An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3
hq_spline	4

class SpectraVoice.**HarmonicType**
An enumeration.

Name	Value
hsin	0
rect	1
org1	2
org2	3
org3	4
org4	5
sin	6
random	7
triangle1	8
triangle2	9
overtones1	10
overtones2	11
overtones3	12
overtones4	13

8.1.9 Vorbis player [TBW]

Controller reference

class `rv.modules.vorbisplayer.`**`VorbisPlayer`**(***kwargs*)
"Vorbis player" SunVox Synth Module

Behaviors:

- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..512>	256
`02` (2)	original_speed	<class 'bool'>	True
`03` (3)	finetune	<Range -128..128>	0
`04` (4)	transpose	<Range -128..128>	0
`05` (5)	interpolation	<class 'bool'>	True
`06` (6)	polyphony_ch	<Range 1..4>	1
`07` (7)	repeat	<class 'bool'>	False

8.2 Effects

8.2.1 Amplifier [TBW]

Controller reference

class `rv.modules.amplifier.`**`Amplifier`**(***kw*)
"Amplifier" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..1024>	256
02 (2)	panning	<Range -128..128>	0
03 (3)	dc_offset	<Range -128..128>	0
04 (4)	inverse	<class 'bool'>	False
05 (5)	stereo_width	<Range 0..256>	128
06 (6)	absolute	<class 'bool'>	False
07 (7)	fine_volume	<Range 0..32768>	32768

8.2.2 Compressor [TBW]

Controller reference

class rv.modules.compressor.**Compressor**(**kw*)
"Compressor" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..512>	256
02 (2)	threshold	<Range 0..512>	256
03 (3)	slope_pct	<Range 0..200>	100
04 (4)	attack_ms	<Range 0..500>	1
05 (5)	release_ms	<Range 1..1000>	300
06 (6)	mode	<enum 'Mode'>	<Mode.peak: 0>
07 (7)	sidechain_input	<Range 0..32>	0

class Compressor.**Mode**
An enumeration.

Name	Value
peak	0
rms	1

8.2.3 DC Blocker [TBW]

Controller reference

class rv.modules.dcblocker.**DcBlocker**(**kw*)
"DC Blocker" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	channels	<enum 'Channels'>	<Channels.stereo: 0>

class DcBlocker.**Channels**
An enumeration.

Name	Value
stereo	0
mono	1

8.2.4 Delay [TBW]

Controller reference

class rv.modules.delay.**Delay**(***kw*)
"Delay" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	dry	<Range 0..512>	256
02 (2)	wet	<Range 0..512>	256
03 (3)	delay_l	<Range 0..256>	128
04 (4)	delay_r	<Range 0..256>	160
05 (5)	volume_l	<Range 0..256>	256
06 (6)	volume_r	<Range 0..256>	256
07 (7)	channels	<enum 'Channels'>	<Channels.stereo: 0>
08 (8)	inverse	<class 'bool'>	False
09 (9)	delay_units	<enum 'DelayUnits'>	<DelayUnits.sec_16384: 0>

class Delay.**Channels**
An enumeration.

Name	Value
stereo	0
mono	1

class Delay.**DelayUnits**
An enumeration.

Name	Value
sec_16384	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

8.2.5 Distortion [TBW]

Controller reference

class `rv.modules.distortion.`**`Distortion`**`(`***kw*`)`
"Distortion" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..256>	128
`02` (2)	type	<enum 'Type'>	<Type.lim: 0>
`03` (3)	power	<Range 0..256>	0
`04` (4)	bit_depth	<Range 1..16>	16
`05` (5)	freq_hz	<Range 0..44100>	44100
`06` (6)	noise	<Range 0..256>	0

class `Distortion.`**`Type`**
An enumeration.

Name	Value
lim	0
sat	1

8.2.6 Echo [TBW]

Controller reference

class `rv.modules.echo.`**`Echo`**`(`***kw*`)`
"Echo" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	dry	<Range 0..256>	256
`02` (2)	wet	<Range 0..256>	128
`03` (3)	feedback	<Range 0..256>	128
`04` (4)	delay	<Range 0..256>	256
`05` (5)	channels	<enum 'Channels'>	<Channels.stereo: 1>
`06` (6)	delay_units	<enum 'DelayUnits'>	<DelayUnits.sec_256: 0>

class `Echo.`**`Channels`**
An enumeration.

Name	Value
mono	0
stereo	1

class Echo.**DelayUnits**
An enumeration.

Name	Value
sec_256	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

8.2.7 EQ [TBW]

Controller reference

class rv.modules.eq.**Eq**(**kw)
"EQ" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	low	<Range 0..512>	256
02 (2)	middle	<Range 0..512>	256
03 (3)	high	<Range 0..512>	256
04 (4)	channels	<enum 'Channels'>	<Channels.stereo: 0>

class Eq.**Channels**
An enumeration.

Name	Value
stereo	0
mono	1

8.2.8 Filter [TBW]

Controller reference

class rv.modules.filter.**Filter**(**kw)
"Filter" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..256>	256
02 (2)	freq	<Range 0..14000>	14000
03 (3)	resonance	<Range 0..1530>	0
04 (4)	type	<enum 'Type'>	<Type.lp: 0>
05 (5)	response	<Range 0..256>	8
06 (6)	mode	<enum 'Mode'>	<Mode.hq: 0>
07 (7)	impulse	<Range 0..14000>	0
08 (8)	mix	<Range 0..256>	256
09 (9)	lfo_freq	<Range 0..1024>	8
0a (10)	lfo_amp	<Range 0..256>	0
0b (11)	set_lfo_phase	<Range 0..256>	0
0c (12)	exponential_freq	<class 'bool'>	False
0d (13)	roll_off	<enum 'RollOff'>	<RollOff.db_12: 0>
0e (14)	lfo_freq_unit	<enum 'LfoFreqUnit'>	<LfoFreqUnit.hz_0_02: 0>
0f (15)	lfo_waveform	<enum 'LfoWaveform'>	<LfoWaveform.sin: 0>

class Filter.**Type**
An enumeration.

Name	Value
lp	0
hp	1
bp	2
notch	3

class Filter.**Mode**
An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3

class Filter.**RollOff**
An enumeration.

Name	Value
db_12	0
db_24	1
db_36	2
db_48	3

class Filter.**LfoFreqUnit**
An enumeration.

Name	Value
hz_0_02	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

class Filter.**LfoWaveform**
An enumeration.

Name	Value
sin	0
saw	1
saw2	2
square	3
random	4

8.2.9 Filter Pro [TBW]

Controller reference

class `rv.modules.filterpro.`**`FilterPro`**(***kw*)
"Filter Pro" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..32768>	32768
`02` (2)	type	<enum 'Type'>	<Type.lp: 0>
`03` (3)	freq_hz	<Range 0..22000>	22000
`04` (4)	freq_finetune	<Range -1000..1000>	0
`05` (5)	freq_scale	<Range 0..200>	100
`06` (6)	exponential_freq	<class 'bool'>	False
`07` (7)	q	<Range 0..32768>	16384
`08` (8)	gain	<Range -16384..16384>	0
`09` (9)	roll_off	<enum 'RollOff'>	<RollOff.db_12: 0>
`0a` (10)	response	<Range 0..1000>	250
`0b` (11)	mode	<enum 'Mode'>	<Mode.stereo: 0>
`0c` (12)	mix	<Range 0..32768>	32768
`0d` (13)	lfo_freq	<Range 0..1024>	8
`0e` (14)	lfo_amp	<Range 0..32768>	0
`0f` (15)	lfo_waveform	<enum 'LfoWaveform'>	<LfoWaveform.sin: 0>
`10` (16)	set_lfo_phase	<Range 0..256>	0
`11` (17)	lfo_freq_unit	<enum 'LfoFreqUnit'>	<LfoFreqUnit.hz_0_02: 0>

class `FilterPro.`**`Type`**
An enumeration.

Name	Value
lp	0
hp	1
bp_const_skirt_gain	2
bp_const_peak_gain	3
notch	4
all_pass	5
peaking	6
low_shelf	7
high_shelf	8

class FilterPro.**RollOff**
An enumeration.

Name	Value
db_12	0
db_24	1
db_36	2
db_48	3

class FilterPro.**Mode**
An enumeration.

Name	Value
stereo	0
mono	1

class FilterPro.**LfoWaveform**
An enumeration.

Name	Value
sin	0
saw	1
saw2	2
square	3
random	4

class FilterPro.**LfoFreqUnit**
An enumeration.

Name	Value
hz_0_02	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

8.2.10 Flanger [TBW]

Controller reference

class rv.modules.flanger.**Flanger**(**_kw_)
"Flanger" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	dry	<Range 0..256>	256
02 (2)	wet	<Range 0..256>	128
03 (3)	feedback	<Range 0..256>	128
04 (4)	delay	<Range 0..1000>	200
05 (5)	response	<Range 0..256>	2
06 (6)	lfo_freq	<Range 0..512>	8
07 (7)	lfo_amp	<Range 0..256>	32
08 (8)	lfo_waveform	<enum 'LfoWaveform'>	<LfoWaveform.hsin: 0>
09 (9)	set_lfo_phase	<Range 0..256>	0
0a (10)	lfo_freq_unit	<enum 'LfoFreqUnit'>	<LfoFreqUnit.hz_0_05: 0>

class Flanger.**LfoWaveform**
An enumeration.

Name	Value
hsin	0
sin	1

class Flanger.**LfoFreqUnit**
An enumeration.

Name	Value
hz_0_05	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

8.2.11 LFO [TBW]

Controller reference

class rv.modules.lfo.**Lfo**(***kw*)
"LFO" SunVox Effect Module

Behaviors:

- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..512>	256
02 (2)	type	<enum 'Type'>	<Type.amplitude: 0>
03 (3)	amplitude	<Range 0..256>	256
04 (4)	freq	<Range 0..2048>	256
05 (5)	waveform	<enum 'Waveform'>	<Waveform.sin: 0>
06 (6)	set_phase	<Range 0..256>	0
07 (7)	channels	<enum 'Channels'>	<Channels.stereo: 0>
08 (8)	frequency_unit	<enum 'FrequencyUnit'>	<FrequencyUnit.hz_64: 0>
09 (9)	duty_cycle	<Range 0..256>	128
0a (10)	generator	<class 'bool'>	False

class `Lfo.`**`Type`**
An enumeration.

Name	Value
amplitude	0
panning	1

class `Lfo.`**`Waveform`**
An enumeration.

Name	Value
sin	0
square	1
sin2	2
saw	3
saw2	4
random	5
triangle	6

class `Lfo.`**`Channels`**
An enumeration.

Name	Value
stereo	0
mono	1

class `Lfo.`**`FrequencyUnit`**
An enumeration.

Name	Value
hz_64	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

8.2.12 Loop [TBW]

Controller reference

class `rv.modules.loop.`**`Loop`**`(`**kw*`)`
"Loop" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..256>	256
`02` (2)	delay	<Range 0..256>	256
`03` (3)	channels	<enum 'Channels'>	<Channels.stereo: 1>
`04` (4)	repeats	<Range 0..64>	0
`05` (5)	mode	<enum 'Mode'>	<Mode.normal: 0>

class `Loop.`**`Channels`**
An enumeration.

Name	Value
mono	0
stereo	1

8.2.13 Modulator [TBW]

Controller reference

class `rv.modules.modulator.`**`Modulator`**`(`*`**kw`*`)`
"Modulator" SunVox Effect Module

Behaviors:

- receives_audio
- receives_modulator
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..512>	256
`02` (2)	modulation_type	<enum 'ModulationType'>	<ModulationType.amplitude: 0>
`03` (3)	channels	<enum 'Channels'>	<Channels.stereo: 0>

class `Modulator.`**`ModulationType`**
An enumeration.

Name	Value
amplitude	0
phase	1
phase_abs	2

class `Modulator.`**`Channels`**
An enumeration.

Name	Value
stereo	0
mono	1

8.2.14 Pitch Shifter [TBW]

Controller reference

class `rv.modules.pitchshifter.`**`PitchShifter`**`(`*`**kw`*`)`
"Pitch shifter" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..512>	256
02 (2)	pitch	<Range -600..600>	0
03 (3)	pitch_scale	<Range 0..200>	100
04 (4)	feedback	<Range 0..256>	0
05 (5)	grain_size	<Range 0..256>	64
06 (6)	mode	<enum 'Mode'>	<Mode.hq: 0>

class PitchShifter.**Mode**
An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3

8.2.15 Reverb [TBW]

Controller reference

class rv.modules.reverb.**Reverb**(**kw*)
"Reverb" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	dry	<Range 0..256>	256
02 (2)	wet	<Range 0..256>	64
03 (3)	feedback	<Range 0..256>	256
04 (4)	damp	<Range 0..256>	128
05 (5)	stereo_width	<Range 0..256>	256
06 (6)	freeze	<class 'bool'>	False
07 (7)	mode	<enum 'Mode'>	<Mode.hq: 0>
08 (8)	all_pass_filter	<class 'bool'>	True
09 (9)	room_size	<Range 0..128>	16
0a (10)	random_seed	<Range 0..32768>	0

class Reverb.**Mode**
An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3

8.2.16 Vibrato [TBW]

Controller reference

class rv.modules.vibrato.**Vibrato**(**kw)
"Vibrato" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..256>	256
02 (2)	amplitude	<Range 0..256>	16
03 (3)	freq	<Range 1..2048>	256
04 (4)	channels	<enum 'Channels'>	<Channels.stereo: 0>
05 (5)	set_phase	<Range 0..256>	0
06 (6)	frequency_unit	<enum 'FrequencyUnit'>	<FrequencyUnit.hz_64: 0>
07 (7)	exponential_amplitude	<class 'bool'>	False

class Vibrato.**Channels**
An enumeration.

Name	Value
stereo	0
mono	1

class Vibrato.**FrequencyUnit**
An enumeration.

Name	Value
hz_64	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

8.2.17 Vocal Filter [TBW]

Controller reference

class rv.modules.vocalfilter.**VocalFilter**(**kw)
"Vocal filter" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..512>	256
02 (2)	formant_width_hz	<Range 0..256>	128
03 (3)	intensity	<Range 0..256>	128
04 (4)	formants	<Range 1..5>	5
05 (5)	vowel	<Range 0..256>	0
06 (6)	voice_type	<enum 'VoiceType'>	<VoiceType.soprano: 0>
07 (7)	channels	<enum 'Channels'>	<Channels.stereo: 0>

class VocalFilter.**VoiceType**
An enumeration.

Name	Value
soprano	0
alto	1
tenor	2
bass	3

class VocalFilter.**Channels**
An enumeration.

Name	Value
stereo	0
mono	1

8.2.18 WaveShaper [TBW]

Controller reference

class rv.modules.waveshaper.**WaveShaper**(***kwargs*)
"WaveShaper" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	input_volume	<Range 0..512>	256
02 (2)	mix	<Range 0..256>	256
03 (3)	output_volume	<Range 0..512>	256
04 (4)	symmetric	<class 'bool'>	True
05 (5)	mode	<enum 'Mode'>	<Mode.hq: 0>
06 (6)	dc_blocker	<class 'bool'>	True

class WaveShaper.**Mode**
An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3

8.3 Misc

8.3.1 Feedback [TBW]

Controller reference

class `rv.modules.feedback.`**`Feedback`**`(`***kw*`)`
"Feedback" SunVox Misc Module

Behaviors:

- receives_audio
- receives_feedback
- sends_audio
- sends_feedback

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..10000>	1000
`02` (2)	channels	<enum 'Channels'>	<Channels.stereo: 0>

class `Feedback.`**`Channels`**
An enumeration.

Name	Value
stereo	0
mono	1

8.3.2 Glide [TBW]

Controller reference

class `rv.modules.glide.`**`Glide`**`(`***kw*`)`
"Glide" SunVox Misc Module

Behaviors:

- receives_notes
- sends_notes

Controllers:

Number	Name	Type	Default
`01` (1)	response	<Range 0..1000>	500
`02` (2)	sample_rate_hz	<Range 1..32768>	150
`03` (3)	offset_on_1st_note	<class 'bool'>	False
`04` (4)	polyphony	<class 'bool'>	True
`05` (5)	pitch	<Range -600..600>	0
`06` (6)	pitch_scale	<Range 0..200>	100
`07` (7)	reset	<class 'bool'>	False

8.3.3 GPIO [TBW]

Controller reference

class `rv.modules.gpio.`**`Gpio`**`(`***kw*`)`
"GPIO" SunVox Misc Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	out	<class 'bool'>	False
`02` (2)	out_pin	<Range 0..64>	0
`03` (3)	out_threshold	<Range 0..100>	50
`04` (4)	in_	<class 'bool'>	False
`05` (5)	in_pin	<Range 0..64>	0
`06` (6)	in_note	<Range 0..128>	0
`07` (7)	in_amplitude	<Range 0..100>	100

8.3.4 MetaModule [TBW]

Controller reference

class `rv.modules.metamodule.`**`MetaModule`**`(`***kwargs*`)`
"MetaModule" SunVox Misc Module

In addition to standard controllers, you can assign zero or more user-defined controllers which map to module/controller pairs in the project embedded within the MetaModule.

Behaviors:

- receives_audio
- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..1024>	256
`02` (2)	input_module	<Range 1..256>	1
`03` (3)	play_patterns	<class 'bool'>	False
`04` (4)	bpm	<Range 1..800>	125
`05` (5)	tpl	<Range 1..31>	6
`06` (6)	user_defined_1	<Range 0..32768>	0
`07` (7)	user_defined_2	<Range 0..32768>	0
`08` (8)	user_defined_3	<Range 0..32768>	0
`09` (9)	user_defined_4	<Range 0..32768>	0
`0a` (10)	user_defined_5	<Range 0..32768>	0
`0b` (11)	user_defined_6	<Range 0..32768>	0

Continued on next page

Table 8.1 – continued from previous page

Number	Name	Type	Default
0c (12)	user_defined_7	<Range 0..32768>	0
0d (13)	user_defined_8	<Range 0..32768>	0
0e (14)	user_defined_9	<Range 0..32768>	0
0f (15)	user_defined_10	<Range 0..32768>	0
10 (16)	user_defined_11	<Range 0..32768>	0
11 (17)	user_defined_12	<Range 0..32768>	0
12 (18)	user_defined_13	<Range 0..32768>	0
13 (19)	user_defined_14	<Range 0..32768>	0
14 (20)	user_defined_15	<Range 0..32768>	0
15 (21)	user_defined_16	<Range 0..32768>	0
16 (22)	user_defined_17	<Range 0..32768>	0
17 (23)	user_defined_18	<Range 0..32768>	0
18 (24)	user_defined_19	<Range 0..32768>	0
19 (25)	user_defined_20	<Range 0..32768>	0
1a (26)	user_defined_21	<Range 0..32768>	0
1b (27)	user_defined_22	<Range 0..32768>	0
1c (28)	user_defined_23	<Range 0..32768>	0
1d (29)	user_defined_24	<Range 0..32768>	0
1e (30)	user_defined_25	<Range 0..32768>	0
1f (31)	user_defined_26	<Range 0..32768>	0
20 (32)	user_defined_27	<Range 0..32768>	0

8.3.5 MultiCtl [TBW]

Controller reference

class rv.modules.multictl.**MultiCtl**(***kwargs*)
"MultiCtl" SunVox Misc Module

Behaviors:

- sends_controls

Controllers:

Number	Name	Type	Default
01 (1)	value	<Range 0..32768>	0
02 (2)	gain	<Range 0..1024>	256
03 (3)	quantization	<Range 0..32768>	32768
04 (4)	out_offset	<Range -16384..16384>	0

8.3.6 MultiSynth [TBW]

Controller reference

class rv.modules.multisynth.**MultiSynth**(***kwargs*)
"MultiSynth" SunVox Misc Module

Behaviors:

- receives_notes

- sends_notes

Controllers:

Number	Name	Type	Default
01 (1)	transpose	<CompactRange -128..128>	0
02 (2)	random_pitch	<Range 0..4096>	0
03 (3)	velocity	<Range 0..256>	256
04 (4)	finetune	<Range -256..256>	0
05 (5)	random_phase	<Range 0..32768>	0
06 (6)	random_velocity	<Range 0..32768>	0
07 (7)	phase	<Range 0..32768>	0
08 (8)	curve2_influence	<Range 0..256>	256

8.3.7 Pitch2Ctl [TBW]

Controller reference

class rv.modules.pitch2ctl.**Pitch2Ctl**(***kw*)
"Pitch2Ctl" SunVox Misc Module

Behaviors:

- receives_notes
- sends_controls

Controllers:

Number	Name	Type	Default
01 (1)	mode	<enum 'Mode'>	<Mode.frequency_hz: 0>
02 (2)	note_off_action	<enum 'NoteOffAction'>	<NoteOffAction.do_nothing: 0>
03 (3)	first_note	<Range 0..256>	0
04 (4)	number_of_semitones	<Range 0..256>	120
05 (5)	out_min	<Range 0..32768>	0
06 (6)	out_max	<Range 0..32768>	32768
07 (7)	out_controller	<Range 0..32>	0

class Pitch2Ctl.**Mode**
An enumeration.

Name	Value
frequency_hz	0
pitch	1

class Pitch2Ctl.**NoteOffAction**
An enumeration.

Name	Value
do_nothing	0
pitch_down	1
pitch_up	2

8.3.8 Sound2Ctl [TBW]

Controller reference

class rv.modules.sound2ctl.**Sound2Ctl**(**kw)
"Sound2Ctl" SunVox Misc Module

Behaviors:

- receives_audio
- sends_controls

Controllers:

Number	Name	Type	Default
01 (1)	sample_rate_hz	<Range 1..32768>	50
02 (2)	channels	<enum 'Channels'>	<Channels.mono: 0>
03 (3)	absolute	<class 'bool'>	True
04 (4)	gain	<Range 0..1024>	256
05 (5)	smooth	<Range 0..256>	128
06 (6)	mode	<enum 'Mode'>	<Mode.hq: 1>
07 (7)	out_min	<Range 0..32768>	0
08 (8)	out_max	<Range 0..32768>	32768
09 (9)	out_controller	<Range 0..32>	0

class Sound2Ctl.**Channels**
An enumeration.

Name	Value
mono	0
stereo	1

class Sound2Ctl.**Mode**
An enumeration.

Name	Value
lq	0
hq	1

8.3.9 Velocity2Ctl [TBW]

Controller reference

class rv.modules.velocity2ctl.**Velocity2Ctl**(**kw)
"Velocity2Ctl" SunVox Misc Module

Behaviors:

- receives_notes
- sends_controls

Controllers:

Number	Name	Type	Default
01 (1)	note_off_action	<enum 'NoteOffAction'>	<NoteOffAction.do_nothing: 0>
02 (2)	out_min	<Range 0..32768>	0
03 (3)	out_max	<Range 0..32768>	32768
04 (4)	out_offset	<Range -16384..16384>	0
05 (5)	out_controller	<Range 0..32>	0

class Velocity2Ctl.**NoteOffAction**

An enumeration.

Name	Value
do_nothing	0
vel_down	1
vel_up	2

CHAPTER 9

Note effect reference [TBW]

CHAPTER 10

Platform-specific features [TBW]

10.1 Android [TBW]

10.2 iOS [TBW]

10.3 Linux [TBW]

10.4 macOS [TBW]

10.5 Maemo [TBW]

10.6 Meego [TBW]

10.7 PalmOS [TBW]

10.8 Raspberry Pi [TBW]

10.9 Windows [TBW]

10.10 Windows CE [TBW]

CHAPTER 11

Developers [TBW]

11.1 SunVox DLL playback engine

11.1.1 Python [TBW]

11.2 Radiant Voices: modify SunVox files [TBW]

11.3 `cgsv` companion module

11.3.1 Installation

Dependencies

Required

- Python 3.5 (or greater)

Installing from PyPI

Use `pip` to install the latest version published to PyPI:

```
$ pip install sunvox-guide
```

Installing from GitHub

> **Warning:** When you install this version, you may run into code that does not yet work correctly, or code whose APIs don't match what is described in the documentation.
>
> Although the project makes an effort to ensure code in the `master` branch is kept working and consistent with documentation, this may not always be the case.

Use `pip` to install the most recent version in the `master` branch:

```
$ pip install 'https://github.com/metrasynth/sunvox-guide/#egg=sunvox-guide'
```

CHAPTER 12

Appendices [WIP]

12.1 Default keyboard shortcuts [TBW]

12.2 SP (set pitch) tuning tables

These tables show the relationship between notes and their corresponding *SP (Set Pitch)* values.

12.2.1 Standard tuning

The built-in tuning used by standard SunVox notes is an equal temperament tuning, using a 440Hz A4 note.

See also:

- "PR??" thread in SunVox forum

Note	SP value
C0	7800
c0	7700
D0	7600
d0	7500
E0	7400
F0	7300
f0	7200
G0	7100
g0	7000
A0	6f00
a0	6e00
B0	6d00
C1	6c00
c1	6b00

Continued on next page

Table 12.1 – continued from previous page

Note	SP value
D1	6a00
d1	6900
E1	6800
F1	6700
f1	6600
G1	6500
g1	6400
A1	6300
a1	6200
B1	6100
C2	6000
c2	5f00
D2	5e00
d2	5d00
E2	5c00
F2	5b00
f2	5a00
G2	5900
g2	5800
A2	5700
a2	5600
B2	5500
C3	5400
c3	5300
D3	5200
d3	5100
E3	5000
F3	4f00
f3	4e00
G3	4d00
g3	4c00
A3	4b00
a3	4a00
B3	4900
C4	4800
c4	4700
D4	4600
d4	4500
E4	4400
F4	4300
f4	4200
G4	4100
g4	4000
A4	3f00
a4	3e00
B4	3d00
C5	3c00
c5	3b00
D5	3a00

Continued on next page

Table 12.1 – continued from previous page

Note	SP value
d5	3900
E5	3800
F5	3700
f5	3600
G5	3500
g5	3400
A5	3300
a5	3200
B5	3100
C6	3000
c6	2f00
D6	2e00
d6	2d00
E6	2c00
F6	2b00
f6	2a00
G6	2900
g6	2800
A6	2700
a6	2600
B6	2500
C7	2400
c7	2300
D7	2200
d7	2100
E7	2000
F7	1f00
f7	1e00
G7	1d00
g7	1c00
A7	1b00
a7	1a00
B7	1900
C8	1800
c8	1700
D8	1600
d8	1500
E8	1400
F8	1300
f8	1200
G8	1100
g8	1000
A8	0f00
a8	0e00
B8	0d00
C9	0c00
c9	0b00
D9	0a00
d9	0900

Continued on next page

Table 12.1 – continued from previous page

Note	SP value
E9	0800
F9	0700
f9	0600
G9	0500
g9	0400
A9	0300
a9	0200
B9	0100

Contributors to this page

Author(s) Matthew Scott

Reviewer(s) iaon, gigi

12.3 Hex conversion tables

12.3.1 First digit (0-f)

Hex	Decimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
a	10
b	11
c	12
d	13
e	14
f	15

12.3.2 Second digit (00-ff)

Hex	Decimal
00	0
10	16
20	32
30	48
40	64
50	80
60	96
70	112
80	128
90	144
a0	160
b0	176
c0	192
d0	208
e0	224
f0	240
ff	255

12.3.3 Third digit (000-fff)

Hex	Decimal
000	0
100	256
200	512
300	768
400	1024
500	1280
600	1536
700	1792
800	2048
900	2304
a00	2560
b00	2816
c00	3072
d00	3328
e00	3584
f00	3840
fff	4095

12.3.4 Fourth digit (0000-8000)

Hex	Decimal
0000	0
1000	4096
2000	8192
3000	12288
4000	16384
5000	20480
6000	24576
7000	28672
8000	32768

Note: SunVox does not use values higher than 8000 hex.

This allows for decimal values 0 to 32768, or -16384 to 16384 with an exact midpoint of 0.

12.4 Community resources [WIP]

12.4.1 Official websites

SunVox home page http://www.warmplace.ru/soft/sunvox/

Forum http://www.warmplace.ru/forum/

User manual http://www.warmplace.ru/wiki/sunvox:manual_en

12.4.2 Download mirrors

SunVox https://github.com/warmplace/sunvox/

SunVox DLL https://github.com/warmplace/sunvox_dll/

12.4.3 Social media

Facebook group https://www.facebook.com/groups/sunvox/

Reddit https://www.reddit.com/r/sunvox

12.4.4 Chat servers

Discord https://discord.gg/uyrjsvc

IRC irc://irc.esper.net/sunvox

Slack https://warmcommunity-slack-signup.herokuapp.com/

Contributors to this page

Author(s) Matthew Scott

12.5 Introduction to music theory [TBW]

12.6 Introduction to sound design [TBW]

12.7 Metrasynth [TBW]

12.8 Sytorial reference [TBW]

CHAPTER 13

How to contribute to this book [WIP]

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

If you ever need assistance with the process of getting your work written and included in this book, please reach out through your favorite medium listed at *Community resources [WIP]*.

We use GitHub to manage our workflow. If you plan on editing, authoring, or translating, please sign up there if you haven't already.

13.1 Sending feedback

You don't need GitHub to provide feedback, requests, suggestions, or other comments. You can post to the WarmPlace forum thread for this book, or discuss it using one of the chat platforms listed in *Community resources [WIP]*.

If you do use GitHub, you can also send feedback by filing an issue at https://github.com/metrasynth/sunvox-guide/issues.

13.2 Authoring and editing

We recommend that you set up a local environment for authoring and editing. This book makes extensive use of figures and audio samples, which are generated by a documentation builder.

13.2.1 Bug reports for documentation builder

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

13.2.2 Setting up the documentation builder

1. Fork sunvox-guide (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/sunvox-guide
cd sunvox-guide
```

3. Perform one-time setup of third-party packages:

   pip install -r requirements.txt -r docs/requirements.txt

3. Create a branch for local development:

```
git checkout -b name-of-what-you-are-working-on
```

   Now you can make your changes locally.

4. Run this command to build documentation:

```
cd docs
make html
```

   Run this command to automatically rebuild documentation every time you make changes:

```
cd docs
watchmedo shell-command -c "make html" -p '*.rst' -i '_build/*' -w -W -R .
```

   If you need to rebuild all pages, run this:

```
cd docs
rm -rf _build/doctrees
```

5. Open the `sunvox-guide/docs/_build/html/index.html` file in your web browser to review the changes.
6. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-what-you-are-working-on
```

7. Submit a pull request through the GitHub website.

13.2.3 Pull Request Guidelines

When you begin authoring a new section, or making a large change, please show others you are working on that section by doing the following:

1. In your branch, change the top-level heading to indicate `[WIP]` at the end.
2. Commit and push that change to GitHub.
3. Open a pull request for the branch.

This will allow immediate discussion as the author needs it, as well as helping avoid duplicated work.

Once you are finished, another editor will assist with proofreading and will merge the results into the official repository.

Before merging, you should:

1. Ensure all added or changed files render without errors.
2. Add yourself to `AUTHORS.rst`.

CHAPTER 14

List of contributors

Contributors are listed in order of when their first contribution was made.

14.1 Editors

These are people who edit the book for structure, consistency, and completeness.

- Matthew Scott

14.2 Authors

These are people who have written original material for this book.

- Matthew Scott
- Jan Koekepan

14.3 Reviewers

These are people who have reviewed portions of this book, and have offered comments or corrections that are now included.

- iaon
- gigi

14.4 Other Contributors

These are people who have offered material for the book in other ways, such as the authors of forum posts used as reference for this book.

- Alexander Zolotov (a.k.a. NightRadio, the author of SunVox)

CHAPTER 15

Planning

15.1 Proposed Table of Contents

(WORK IN PROGRESS)

- Preface
- **Introduction**
  - Who is this book for?
  - What is SunVox?
  - What is a modular synthesizer?
  - What is a tracker?
  - How to read this book
  - Conventions used in this book
- **Getting Started**
  - **Installing SunVox**
    - (page for each platform)
  - Configuring MIDI inputs (optional)
  - **Your first sounds**
    - Selecting a module
    - Playing notes using the on-screen keyboard
    - Playing notes with a PC keyboard
    - Playing notes with a MIDI controller
  - **Terminology**
    - Project

        - Module
        - Controller
        - Connection
        - Timeline
        - Pattern
        - Row
        - Track
        - Note
        - Velocity
        - Effect
        - XX, YY, and XXYY values
    - **Navigating the Interface**
        - Keyboard
        - Mouse
        - Touch
        - Multitouch
    - **Introduction to hex**
        - How SunVox helps you
    - Creating an empty project
- **Beginners**
    - Adding and connecting modules
    - Changing controller values
    - Creating a pattern in the timeline
    - Editing patterns
    - Adding note effects
    - Recording notes to patterns
    - Moving patterns in the timeline
    - Changing the properties of a pattern
    - Exporting to WAV files
- **Intermediate**
    - Creating your own template project
    - Combining note effects
    - File management
    - Pattern management
    - Clipboards
- **Advanced**

    - **Mastering**
        - Inside SunVox
        - Using external apps
    - Live performance
    - Tuning and microtonal music
    - **BUilding MetaModules**
        - Effects
        - Synths
        - Playing back patterns
        - Switching input
- **Cookbook**
    - (convert forum posts to detailed examples)
    - (analyze clever tricks from example songs and synths)
- **Module reference**
    - **Synths**
        - Analog Generator
        - DrumSynth
        - FM
        - Input
        - Kicker
        - Sampler
        - Spectravoice
        - Vorbis player
    - **Effects**
        - Amplifier
        - Compressor
        - DC Blocker
        - Delay
        - Distortion
        - Echo
        - EQ
        - Filter
        - Filter Pro
        - LFO
        - Loop
        - Modulator

        - Pitch Shifter
        - Reverb
        - Vibrato
        - Vocal filter
        - Wave Shaper
    - **Misc**
        - Glide
        - MultiSynth
        - Feedback
        - GPIO
        - MetaModule
        - MultiCtl
        - Pitch2ctl
        - Sound2ctl
        - Velocity2ctl
- **Note effect reference**
    - 01, 02: Slide up, down
    - 03: Slide to note
    - 04: Vibrato
    - 07, 09: Set sample offset
    - 08: Arpeggio
    - 0A: Slide velocity up/down
    - 0F: Set playing speed
    - 11, 12: Fineslide up, down
    - 13, 14: Bypass/solo/mute
    - 19: Re-trigger during line
    - 1C: Cut note during line
    - 1D: Delay start during line
    - 1F: Set BPM
    - 20: Note probability
    - 21: Note probability (random velocity)
    - 22, 23: Set controller to random value
    - 30: Stop playing
    - 40..5F: Delay event for line fraction
- **Platform-specific features**
    - Linux

CHAPTER 16

Changelog

16.1 0.1.0 (under development)

- Initial release.

CHAPTER 17

Licensing

17.1 *The Complete Guide to SunVox* book license

17.2 `cgsv` package license

CHAPTER 18

Indices and tables

- genindex
- modindex
- search

Index

M

P

R

S

V

W