

# 1 Zynaddsubfx

This documentation is a work in progress

## 1.1 Getting Started

In most cases ZynAddSubFX is used it will not alone, and it will be linked with other programs.

### 1.1.1 MIDI

MIDI can be used to connect other applications to ZynAddSubFX. Under ALSA on Linux the standard tool to connect ZynAddSubFX's MIDI is aconnect.

ZynAddSubFX has one incoming MIDI port. Once another application is connected to this port, then it should be able to send information on notes, pitches, and modulations to ZynAddSubFX. For more specific information see INSERT CHAPTER NAME HERE.

### 1.1.2 JACK

JACK can be used to connect the output of ZynAddSubFX to other applications.

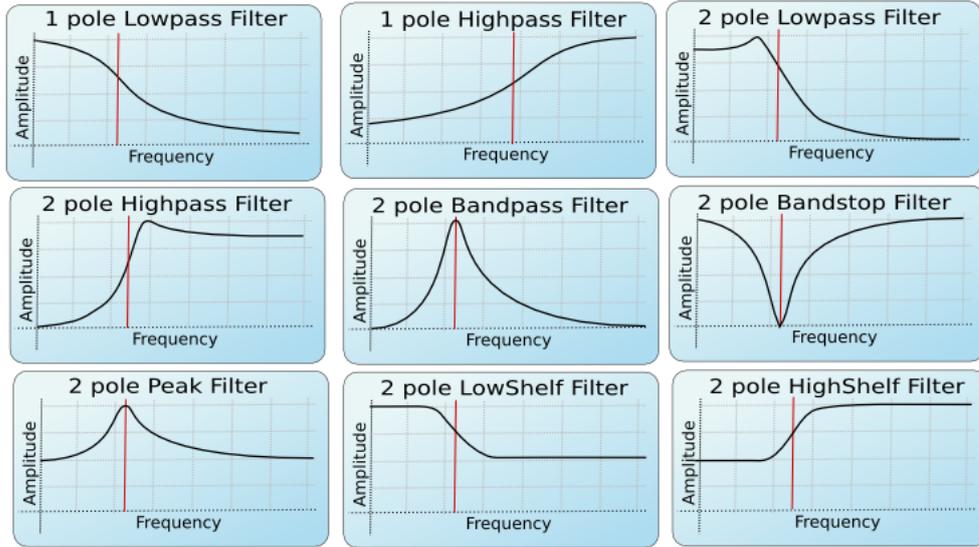
## 1.2 Filters

In ZynAddSubFX filters can be used at several different stages to increase or decrease a defined set of frequencies.

### 1.2.1 Types

ZynAddSubFX currently supports a wide set of filters that it can provide

## ZynAddSubFX filter types

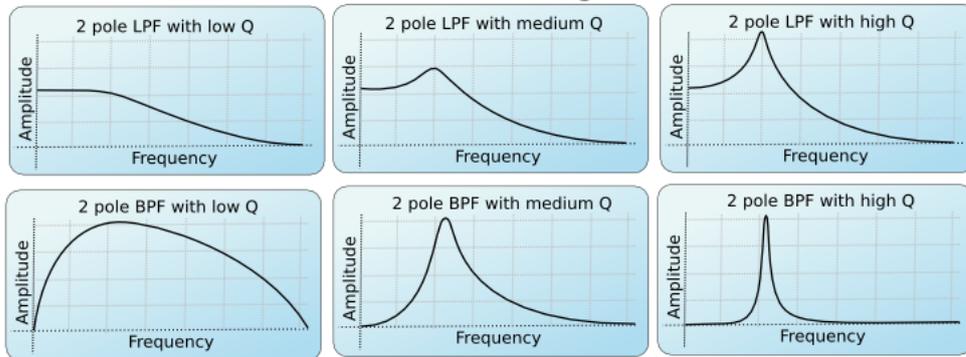


### 1.2.2 Values

Each of these filters will be affected by their Q values, number of poles, and cutoff values.

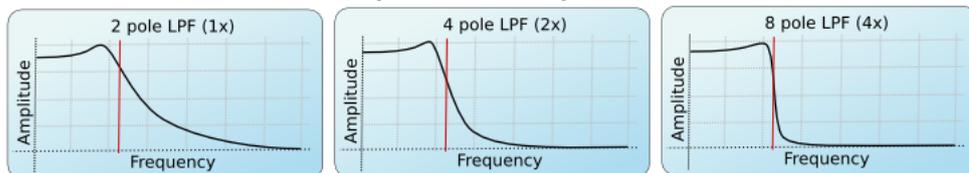
The following diagram shows the frequency response under different Q values:

#### Low Q vs. High Q



The following diagram shows the frequency response under a different number of poles:

#### 2 pole vs. 8 pole



### 1.2.3 User Interface

A picture of the UI will go here

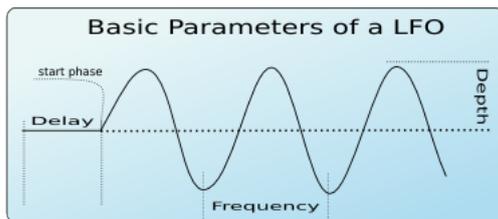
## 1.3 LFO

### 1.3.1 Introduction

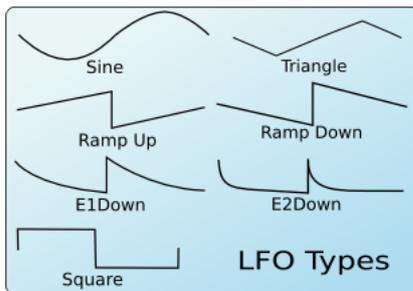
"LFO" means Low Frequency Oscillator. These oscillators are not used to make sounds by themselves, but they change some parameters (like the frequencies, the amplitudes or the filters).

The LFO has some basic parameters:

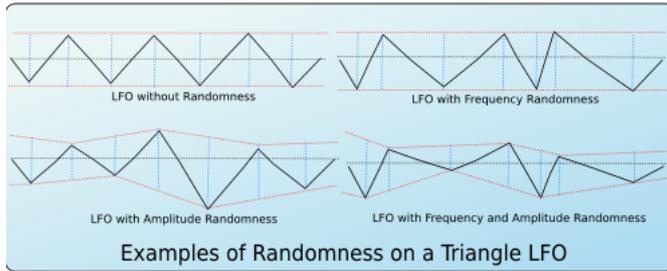
- **Delay:** This parameter sets how much time takes since the start of the note to the start of the LFO
- **Start Phase:** The position that a LFO will start at
- **Frequency:** How fast the LFO is (i.e. how fast the parameter's controlled by the LFO changes)
- **Depth:** The amplitude of the LFO (i.e. how much the parameter's controlled by the LFO changes)



Another important LFO parameter is the shape. There are many LFO Types according to the shape. ZynAddSubFX supports the following LFO shapes:



Another parameter is the LFO Randomness. It modifies the LFO amplitude or the LFO frequency at random. In ZynAddSubFX you can choose how much the LFO frequency or LFO amplitude changes by this parameter. In the following images are shown some examples of randomness and how changes the shape of a triangle LFO.



Other parameters are:

- **Continuous mode:** If this mode is used, the LFO will not start from "zero" on each new note, but it will be continuous. This is very useful if you apply on filters to make interesting sweeps.
- **Stretch:** It controls how much the LFO frequency changes according to the note's frequency. It can vary from negative stretch (the LFO frequency is decreased on higher notes) to zero (the LFO frequency will be the same on all notes) to positive stretch (the LFO frequency will be increased on higher notes).

### 1.3.2 User Interface

In ZynAddSubFX, LFO parameters are shown as:



These parameters are:

- **Freq:** LFO Frequency
- **Depth:** LFO Depth
- **Start:** LFO Start Phase - If this knob is at the lowest value, the LFO Start Phase will be random.
- **Delay:** LFO Delay
- **A.R.:** LFO Amplitude Randomness
- **F.R.:** LFO Frequency Randomness
- **C.:** LFO Continuous Mode
- **Str.:** LFO Stretch - in the image above the LFO stretch is set to zero

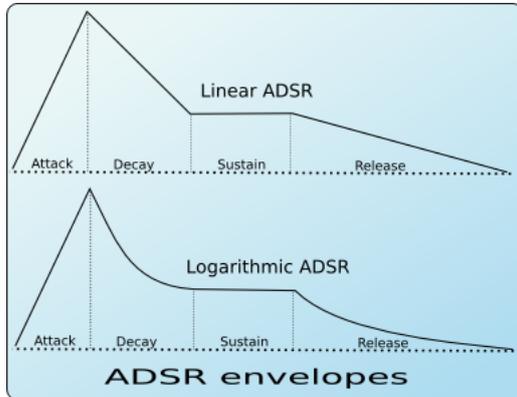
## 1.4 Envelopes

### 1.4.1 Introduction

Envelopes control how the amplitude, the frequency, or the filter changes over time.

### 1.4.2 Amplitude Envelopes

These envelopes controls the amplitude of the sound. In ZynAddSubFX, amplitude envelopes can be linear or logarithmic. In the next image, it is shown the differences between these envelopes.

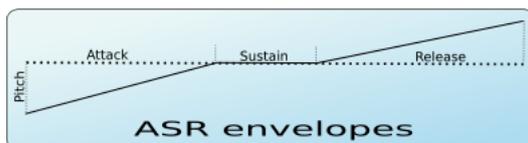


The amplitude envelope is divided into:

- **Attack:** Begins at the Note On. The volume starts from 0 to the maximum. In ZynAddSubFX, the attack is always linear.
- **Decay:** The volume drops from the maximum value to a level called "Sustain level"
- **Sustain:** The volume remains constant until the key is depressed (Note Off). After this, the last stage take place.
- **Release:** The volume drops to zero

### 1.4.3 Frequency Envelopes

These envelopes controls the frequency (more exactly, the pitch) of the oscillators. The following picture draws the stages of these envelopes.



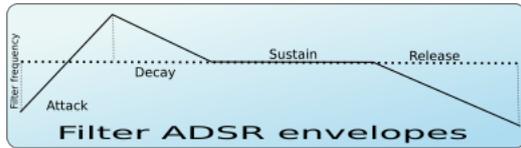
The dotted line represents the real pitch of the sound without the envelope.

The frequency envelopes are divided into 3 stages:

- **Attack:** Begins at the Note On. The frequency starts from a certain value and glides to the real frequency of the note.
- **Sustain:** The frequency is the same on over the sustain period
- **Release:** This stage begins on Note Off and glides the frequency of the note to a certain value

### 1.4.4 Filter Envelopes

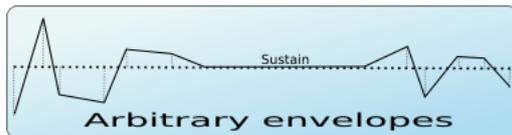
These envelopes controls the cutoff frequency of the filters and are divided into



- **Attack:** Begins at the Note On. The cutoff frequency starts from a certain value and glides to another value
- **Decay:** The cutoff frequency continues to glide to the real cutoff frequency value of the filter (dotted line)
- **Sustain:** the cutoff frequency is the same on over the sustain period (dotted line)
- **Release:** this stage begins on Note Off and glides the filter cutoff frequency of the note to a certain value

### 1.4.5 Freemode Envelopes

For all envelope there is a mode that allows the user to set an arbitrary number of stages and control points. This mode is called Freemode.



Only stage that always remains defined is the Sustain, where the envelopes freezes until a Note Off event.

### 1.4.6 User Interface

All the envelope types has some common controls:

- **E:** Shows a window that you can view the real envelope shape or convert to free mode to edit it
- **Stretch:** How the envelope is stretched according the note. On the higher notes the envelopes are shorter than lower notes. In the leftmost value, the stretch is zero. The rightmost use a stretch of 200%; this means that the envelope is stretched about 4 times/octave.
- **frcR:** Forced release. This means that if this option is turned on, the release will go to the final value, even if the sustain stage is not reached. Usually, this must be set.

The parameters for Amplitude Envelopes for ZynAddSubFX are:



- **A.dt**: Attack duration
- **D.dt**: Decay duration
- **S.Val**: Sustain value
- **R.dt**: Release time
- **L**: If this option is set, the envelope is linear, otherwise, it will be logarithmic

For Frequency Envelopes the interface has the following parameters:



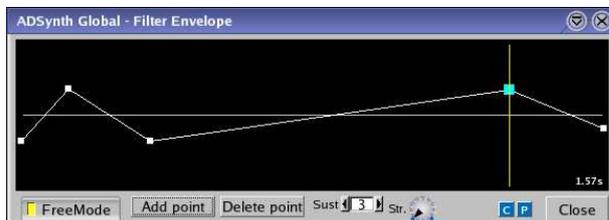
- **A.val**: Attack value
- **A.dt**: Attack duration
- **R.dt**: Release time
- **R.val**: Release value

Filter Envelopes has the parameters:



- **A.val**: Attack value
- **A.dt**: Attack duration
- **D.val**: Decay value
- **D.dt**: Decay time
- **R.dt**: Release time
- **R.val**: Release value

The Freemode envelopes has a separate window to set the parameters and controls:



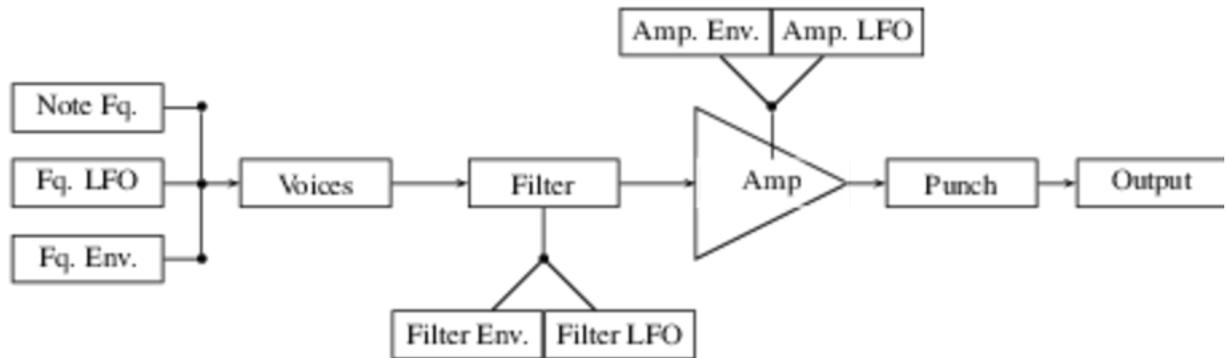
- **Control points:** You can move the points using the mouse. In the right on the windows, it is shown the total duration of the envelope. If the mouse button will be pressed on a control point, it will be shown the duration of the stage where the point is.
- **FreeMode:** this button activates or deactivates the freemode mode.
- **Add Point:** Adds the point next to the current selected point. You can select a point by clicking on it.
- **Delete point:** Removes the point from the envelope.
- **Sust.:** Set the sustain point. It is shown using the yellow line.
- **Str.:** Envelope stretch

## 1.5 Add Synth

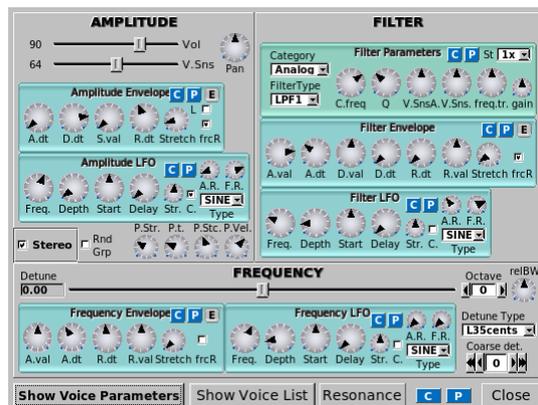
Additive Synthesis is one of the three major synthesis engines available in ZynAddSubFX. Overall it adds together several voices with an oscillator as their sound source.

### 1.5.1 High Level (Global)

From a high level perspective, Add synth can be understood with this block diagram:



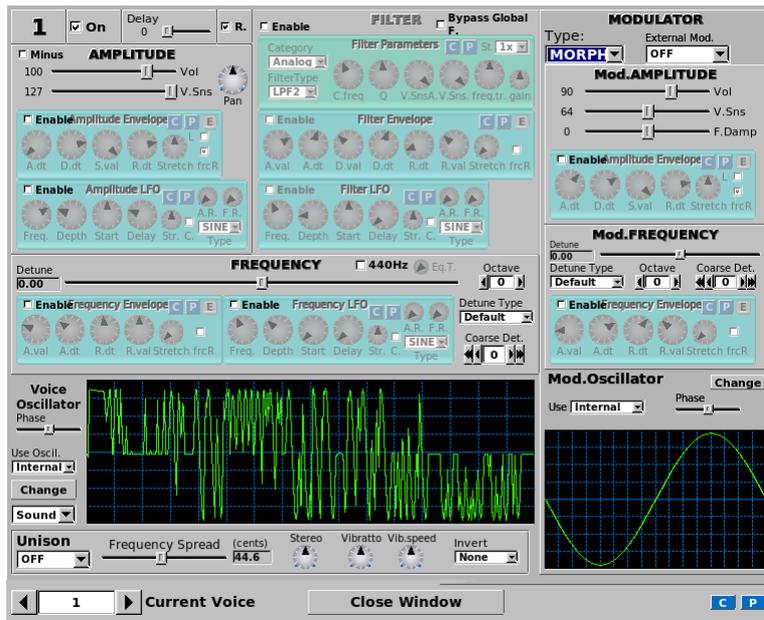
The red/yellow nodes are controllable with the main adsynth window.



The sum of the voices are passed through filters and amplification to produce the final sound. This could lead one to think that ad-note is just a bunch of minor postprocessing and at this level much of the sound generation is hidden.

### 1.5.2 Voices

The voice gives access to a similar setup to the global parameters and then some more, such as the modulator, oscillator, and unison features.



### Modulation

Within the options for modulation, one can select: \* Morph \* Ring Modulation \* Phase Modulation \* Frequency Modulation \* Disabled

### Unison

Unison is useful in creating the chorus like sound of many simultaneous oscillators

### 1.6 Controller



### 1.6.1 General

- **ModWh**: Modulation Wheel depth
- **Exp MWh**: Exponential Modulation Wheel (changes modulation scale to exponential)
- **BwDpth**: Bandwidth Depth
- **Exp BW**: Exponential Bandwidth (changes bandwidth scale to exponential)
- **PanDpth**: Panning Depth
- **FltQ**: Filter Q (resonance) depth
- **FltCut**: Filter Cutoff frequency depth
- **Expr**: enable/disable expression
- **Vol**: enable/disable receiving volume controller
- **FMamp**: enable/disable receiving Modulation Amplitude controller (76)
- **Sustain**: enable/disable sustain pedal
- **PWheelB.Rng (cents)**: Pitch Wheel Bend Range (cents; 100 cents = 1 halftone)

### 1.6.2 Portamento

- **Rcv.**: If the part receives portamento On/Off (65) controller
- **time**: The duration of the portamento
- **thresh**: The threshold of the portamento. It represents the minimum or the maximum number of halftones (or hundred cents) required to start the portamento. The difference is computed between the last note and current note.
- **th.type**: The threshold type. Checked means that the portamento activates when the difference of frequencies is above the threshold ("thresh"); not checked is for below the threshold.

---

The threshold refers to the frequencies and not to MIDI notes (you should consider this if you use microtonal scales).

---

#### Proportional Portamento

- **Propt.**: If the portamento is proportional to ratio of frequencies
- **Prp. Rate**: Ratio needed to double the time of portamento
- **Prp. Dpth**: The divergence from

### 1.6.3 Resonance

- **CFdpth**: resonance center controller depth
- **BWdpth**: resonance bandwidth controller depth

## 1.7 Persistence

As with most applications ZynAddSubFX allows for one to save your work and reload it.

### 1.7.1 Saving it all

One of the simplest ways to save your work is to save the entire session. This can be done through the File menu and will result in the creation of an .xmz file. Once created, this file will hold the settings for all settings within that session, such as microtonal tunings, all patches, system effects, insertion effects, etc. . .

### 1.7.2 Saving Parts

In many cases saving everything is not what is desired. Saving a patch later on is one such example.

#### Patches

In order to save a patch, one can either save it from the instruments menu or through the bank window.

With the instrument menu, one can just save the file to any given location with the .xiz extension.

With the banks menu, one can assign a patch to a given slot with a bank. This instrument will remain here for future use until it is deleted. To see the physical location of the .xiz file, one should check the File→Settings→Bank\_Root\_Dirs window to see the paths for banks.

---

You need to have write permissions to add instruments to the bank.

---

#### Presets

Have a favorite setting for an envelope, a difficult to reproduce oscillator? Then presets are for you. Presets allow for one to save the settings for any of the components which

support copy/paste operations. This is done with preset files (.xpz), which get stored in the folders indicated by File→Settings→Preset\_Root\_Dirs.

### 1.7.3 Summary

#### Extension Summary

xmz Everything  
xiz Instrument  
xsx Scale Settings  
xpz Presets

## 1.8 Appendix A: MIDI Defaults

### Default MIDI Connections

001 - Modulation Wheel  
007 - Volume  
010 - Pan  
011 - Expression  
064 - Sustain  
065 - Portamento Enable  
071 - Filter Q  
074 - Filter Cutoff  
075 - Bandwidth(\*)  
076 - Modulation Amplitude(\*)  
077 - Resonance Center Frequency(\*)  
078 - Resonance Bandwidth(\*)  
120 - All Sounds Off  
121 - Reset All Controllers  
123 - All Notes Off

The entries with (\*) are not within the General Midi specification

### .0.1 Introduction to CMake

Note: This section is mostly copied from the OpenSceneGraph wiki, at: [//www.openscenegraph.org/projects/osg/wiki/Build/CMake](http://www.openscenegraph.org/projects/osg/wiki/Build/CMake)

ZynAddSubFX uses CMake as its unified build system. CMake is able to read simple build scripts from the source tree and create from this a platform-specific build system. This build system can be in the form of VisualStudio project files, Unix Makefiles or XCode project files. CMake is able to automatically locate external dependencies, and build options.

The use of a unified build system has allowed to avoid build breakages that were common in the previous build method of maintaining three separate build targets for VisualStudio, Unix "make" and XCode. It also reduces the maintenance burden for core developers and contributors. Taken together usage of CMake should result in better consistency and more stable builds across all platforms for end users and a greater productivity in development of new versions. Hopefully with greater consistency of builds across platforms it will be easier for developers to use the development version of ZynAddSubFX and help contribute to its testing and refinement, leading to a high-quality code base.

## .0.2 Quick start guide

For the impatient ones, here is a quick guide on how to immediately build ZynAddSubFX from source.

■ Note: This assumes that you already have a copy of the source.

```
SPURIOUS TEXT#enter the source directory
cd zynaddsubfx

#make a directory for an out-of-source build
mkdir build
cd build

#generate a cmake build project here from the cmake root, which is
#found in the directory below the current one
cmake ..

#OPTIONAL: Adjust compile variables in the Cache file:
ccmake .

#And finally, build as usual using make
make
```

Usually there are several methods to obtain a copy of ZynAddSubFX.

### SourceForge

[//sourceforge.net/projects/zynaddsubfx/files/](http://sourceforge.net/projects/zynaddsubfx/files/)

### Distribution

apt/yum/others

### Git

[git://zynaddsubfx.git.sourceforge.net/gitroot/zynaddsubfx/zynaddsubfx](http://zynaddsubfx.git.sourceforge.net/gitroot/zynaddsubfx/zynaddsubfx)

## .0.3 Introduction to Git

For those who want to live on the bleeding edge or who want to assist with making sure that the next release has fewer bugs, you will want to get acquainted with git. Git is used

to manage the source code for this project and can be used to quickly and easily get an up-to-date copy of the source code.

## Getting the Source Code

In order to get a copy of the ZynAddSubFX source code, all that needs to be done is:

```
SPURIOUS TEXTgit clone git://zynaddsubfx.git.sourceforge.net/gitroot/
    zynaddsubfx/zynaddsubfx

cd zynaddsubfx

#Download additional resources
git submodule init
git submodule update
```

You should now be in the directory of the source code.

For simple steps on building, please see Appendix B of the manual.

## Checking out a branch

Lets say that development has extended into the creation of a new feature that you want to preview. For the sake of this guide, lets assume that the name of the branch that the feature is on is foo.

```
SPURIOUS TEXT#checkout the foo branch from sourceforge
git checkout --track -b foo origin/foo

#lets checkout the primary branch again
git checkout master

#hop back to the other branch
git checkout foo
```

Now one should be able to change branches and go into the build directory (as described in Appendix B) and recompile ZynAddSubFX.

---

When using branches other than the master be aware that stability may suffer

---