Zynaddsubfx

Paul Nasca and Mark McCurry

Table of Contents

JavaScript must be enabled in your browser to display the table of contents.

This documentation is a work in progress

1. Getting Started

ZynAddSubFX is a fairly complex software synthesizer with a very large number of controls. As such, it is not alway obvious how to use ZynAddSubFX.

Many applications under Linux transport MIDI over ALSA and transmit audio over JACK. ZynAddSubFX can be run in this configuration by running:

```
zynaddsubfx -I alsa -O jack -a
```

This sets the input driver to be alsa and the output driver to be jack, which should attempt to autoconnect to your soundcard as per the *-a* flag. If this is your first time running ZynAddSubFX, you will see a screen that lets you choose between the advanced and beginner interface. Currently the beginner interface is deprecated, so the advanced one is recommended.

Now you should be able to see ZynAddSubFX's main window, from which you can setup patches, effects, and general configurations, but more importatnly it provides links into the parameters of the patches. ZynAddSubFX is a powerful tool with a number of base patches, but its true power lies in the ability to make your own patches.

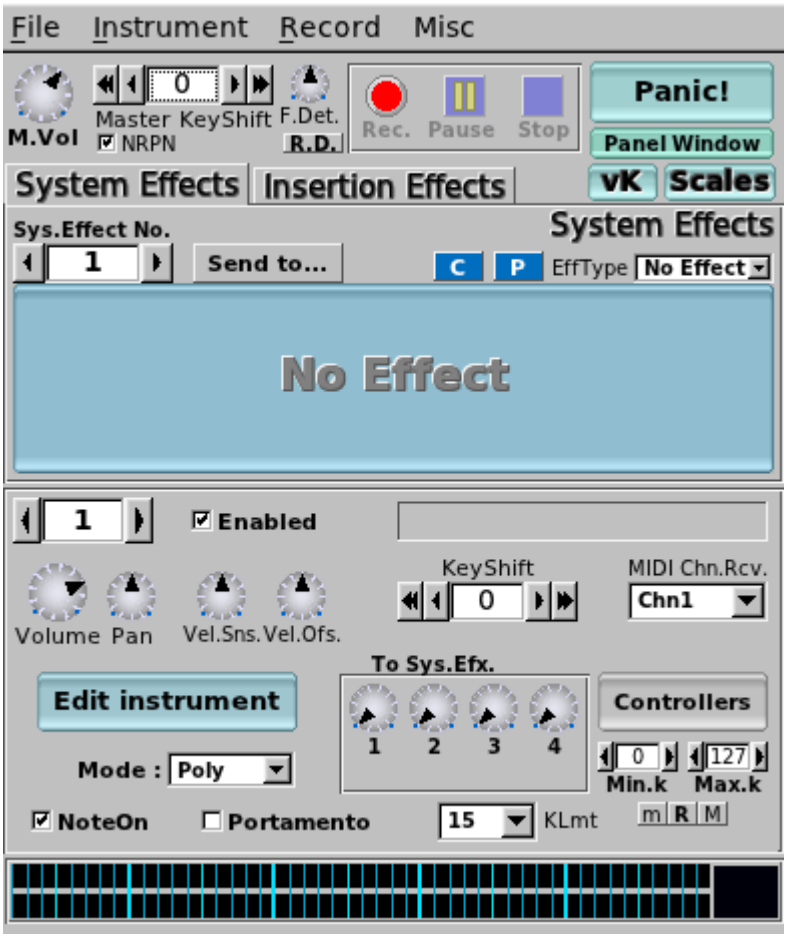


Figure 1. Main Window

For basic usage, you will want to use the button to the right of the enabled label. This button will allow for one to select the desired instrument from the banks that ZynAddSubFX has available. To play notes in ZynAddSubFX, either utilize the builtin virtual keyboard (accessible via the vK button) or connect your keyboard to the system and use **aconnect** to connect it to ZynAddSubFX (assuming that ALSA was used).

This main window provides access to a number of more advanced features. Some of these features are:

- System Effects
- Insertion Effects
- Recording
- Part Settings (instrument level settings)
- Master Settings
- Microtonal Settings

For instance to use the recording feature, a wave file must be selected from the recording menu and then the recording can be started with the record button and stopped with the stop button. This is a simple and quick way of recording some samples from ZynAddSubFX, though there are more full featured options available via JACK recording tools.

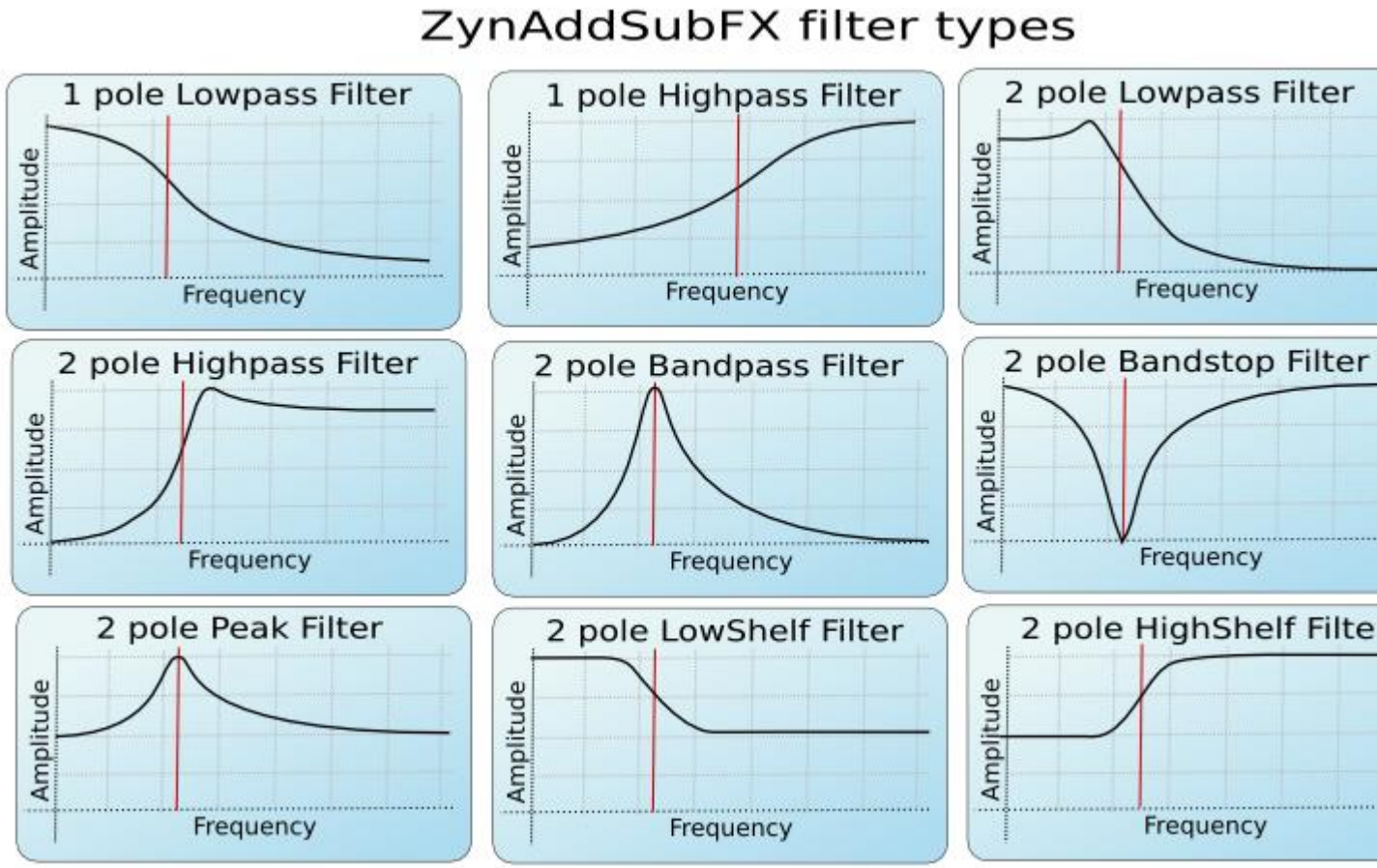
> **Note** After hitting record, the wave file will not start recording until a new key has been pressed via either an external midi source or the virtual keyboard Both system and insertion effects can be accessed, the properties are available as well as properties of each instrument.

2. Filters

ZynAddSubFX offers several different types of filters, which can be used to shape the spectrum of a signal. The primary parameters that affect the characteristics of the filter are the cutoff, resonance, filter stages, and the filter type.

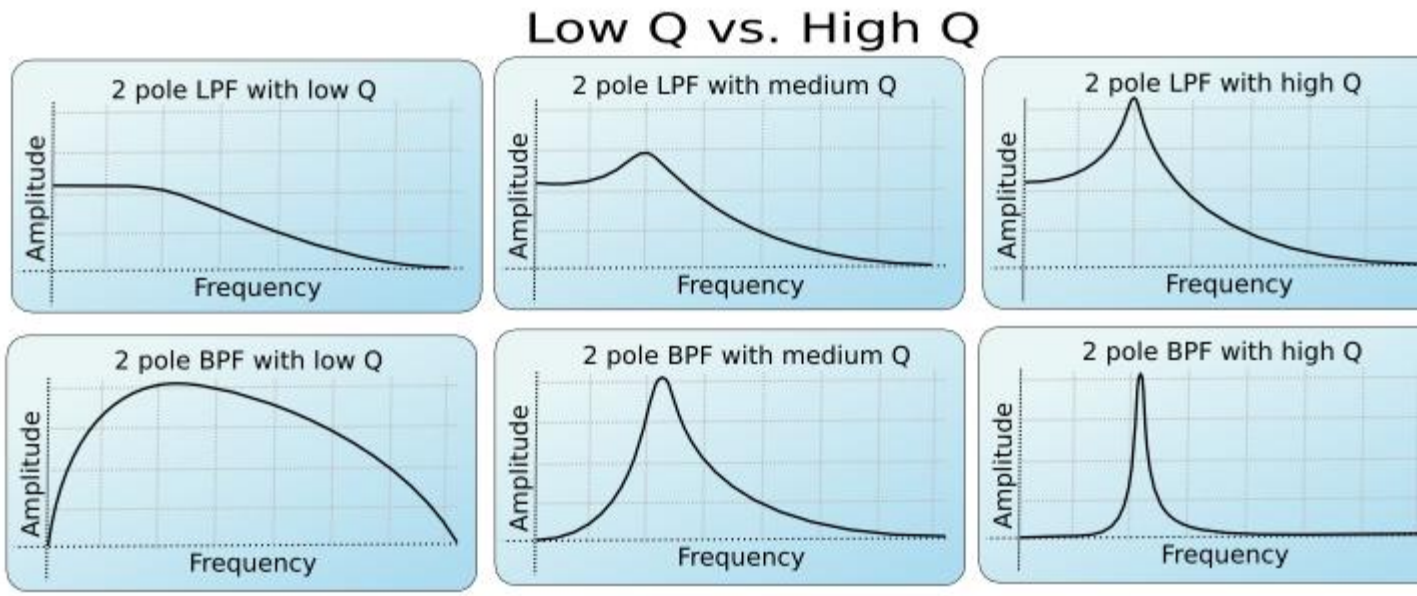
- **Cutoff**: This value determines which frequency marks the changing point for the filter. In a low pass filter, this value marks the point where higher frequencies are attenuated.
- **Resonance**: The resonance of a filter determines how much excess energy is present at the cutoff frequency. In ZynAddSubFX, this is represented by the Q-factor, which is defined to be the cutoff frequency divided by the bandwidth. In other words higher Q values result in a much more narrow resonant spike.
- **Stages**: The number of stages in a given filter describes how sharply it is able to make changes in the frequency response.

The basic *analog* filters that ZynAddSubFX offers are shown below, with the center frequency being marked by the red line. The *state variable* filters should look quite similar.

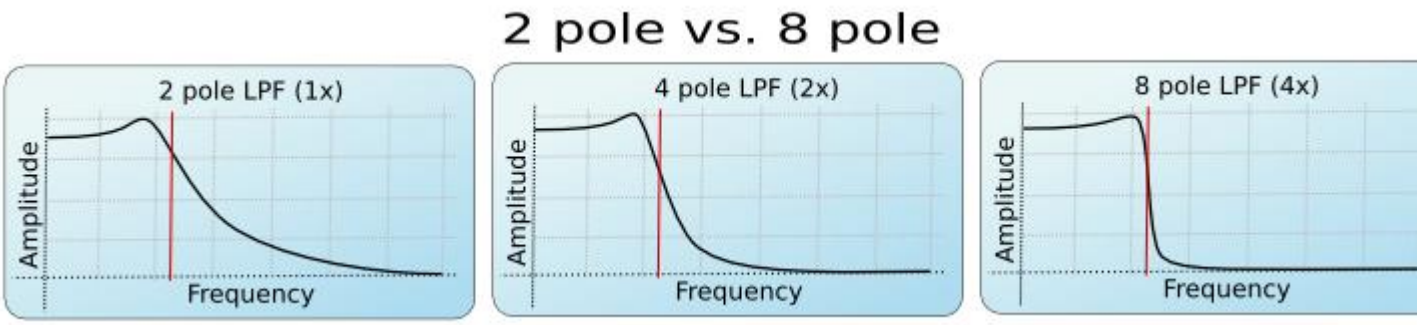


As previously mentioned, the Q value of a filter affects how concentrated the signal's energy is at the cutoff frequency; The result of differing Q values are below.

Tip For many classical analog sounds, high Q values were used on sweeping filters. A simple high Q low pass filter modulated by a strong envelope is usually sufficient to get a good sound.



Lastly, the affect of the order of the filter can be seen below. This is roughly synonymous with the number of stages of the filter. For more complex patches it is important to realize that the extra sharpness in the filter does not come for free as it requires many more calculations being performed; This phenomena is the most visible in subsynth, where it is easy to need several hundred filter stages to produce a given note.



2.1. User Interface

- **C.freq**: Cutoff frequency
- **Q**: Level of resonance for the filter
- **V.SnsA.**: Velocity sensing amount for filter cutoff
- **V.Sns.**: Velocity sensing function

- **freq.tr**: Frequency tracking amount. When this parameter is positive, higher note frequencies shift the filter's cutoff frequency higher.
- **gain**: Additional gain/attenuation for filter
- **St**: Filter stages

> **Note** TODO add a lengthy section on the formant filter setup

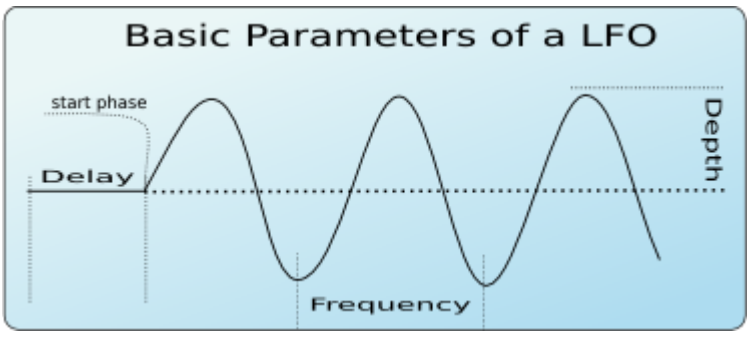
3. LFO

3.1. Introduction

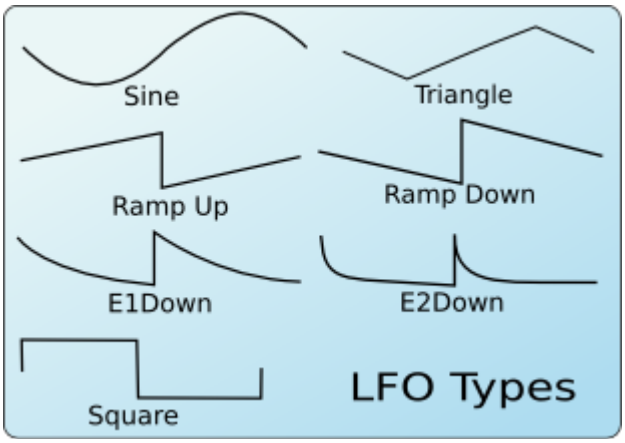
"LFO" means Low Frequency Oscillator. These oscillators are not used to make sounds by themselves, but they changes somes parameters (like the frequencies, the amplitudes or the filters).

The LFOs has some basic parameters:

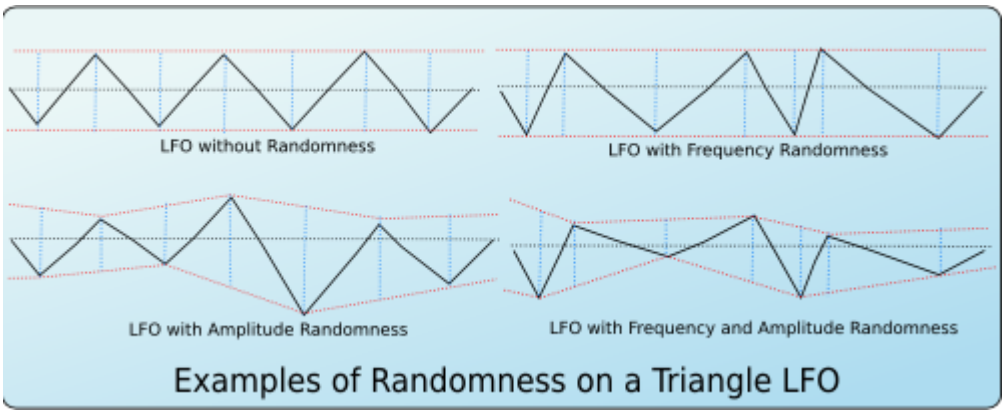
- **Delay**: This parameter sets how much time takes since the start of the note to the start of the LFO
- **Start Phase**: The possition that a LFO will start at
- **Frequency**: How fast the LFO is (i.e. how fast the parameter's controlled by the LFO changes)
- **Depth**: The amplitude of the LFO (i.e. 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 folowing 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 folowing images are shown some examples of randomness and how changes the shape of a triangle LFO.



Other parameters are:

- **Continous 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 usefull if you apply on filters to make interesting sweeps.
- **Stretch**: It controlls 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).

3.2. User Interface

In ZynAddSubFX, LFO parameters are shown as:

Theese 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 Randomnes
- **F.R.**: LFO Frequency Randomness
- **C.**: LFO Continous Mode
- **Str.**: LFO Stretch - in the image above the LFO stretch is set to zero

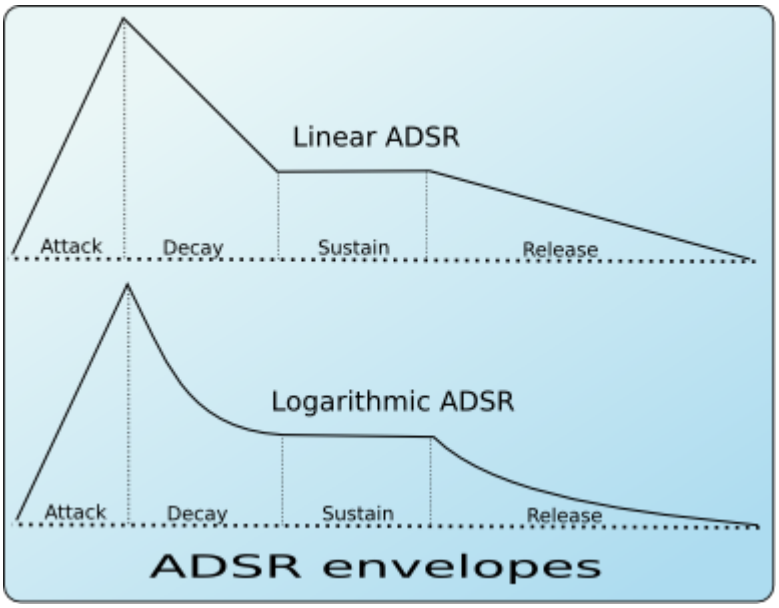
4. Envelopes

4.1. Introduction

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

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.

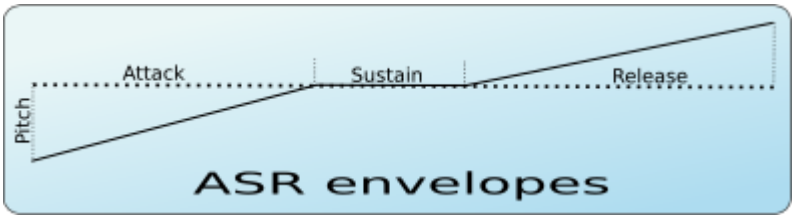


The amplitude envelope is divided into:

- **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

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.



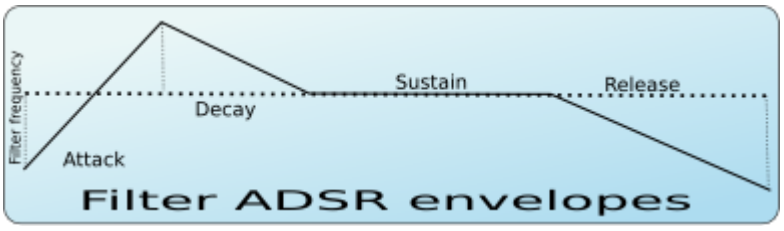
The dotted line represents the real pitch of the sound without the envelope.

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

4.4. Filter Envelopes

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

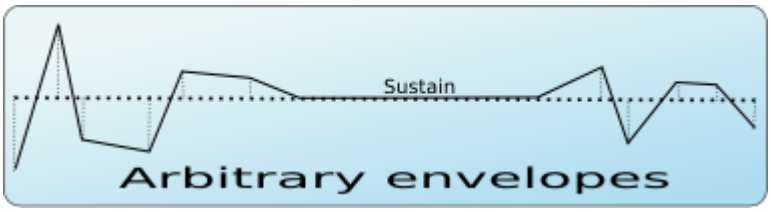


- **Attack**: Begins at the Note On. The cutoff frequency starts from a certain value and glides to another value
- **Decay**: The cutoff frequency continues to glide to the real cutoff frequency value of the filter (dotted line)
- **Sustain**: the cutoff frequency is the same on over the sustain period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

- **Release**: this stage begins on Note Off and glides the filter cutoff frequency of the note to a certain value

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.

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

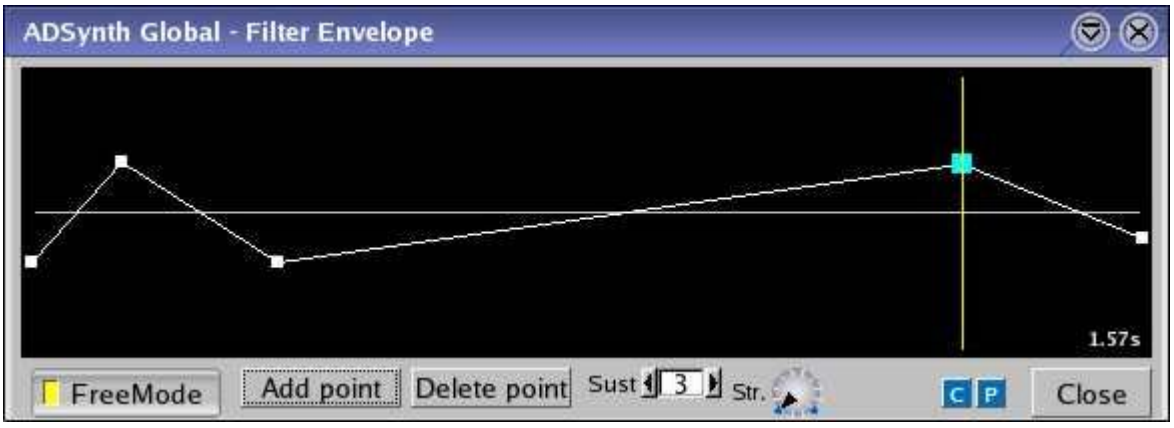
For Frequency Envelopes the interface has the following parameters:

- **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

5. AdSynth

AdSynth, a primarily additive synthesis engine, is one of the three major synthesis engines available in ZynAddSubFX. The basic concept of this engine is the summation of a collection of voices, each of which consist of oscillators.

5.1. High Level (Global)

AdSynth's global level consists of the elements shown in the below figure:

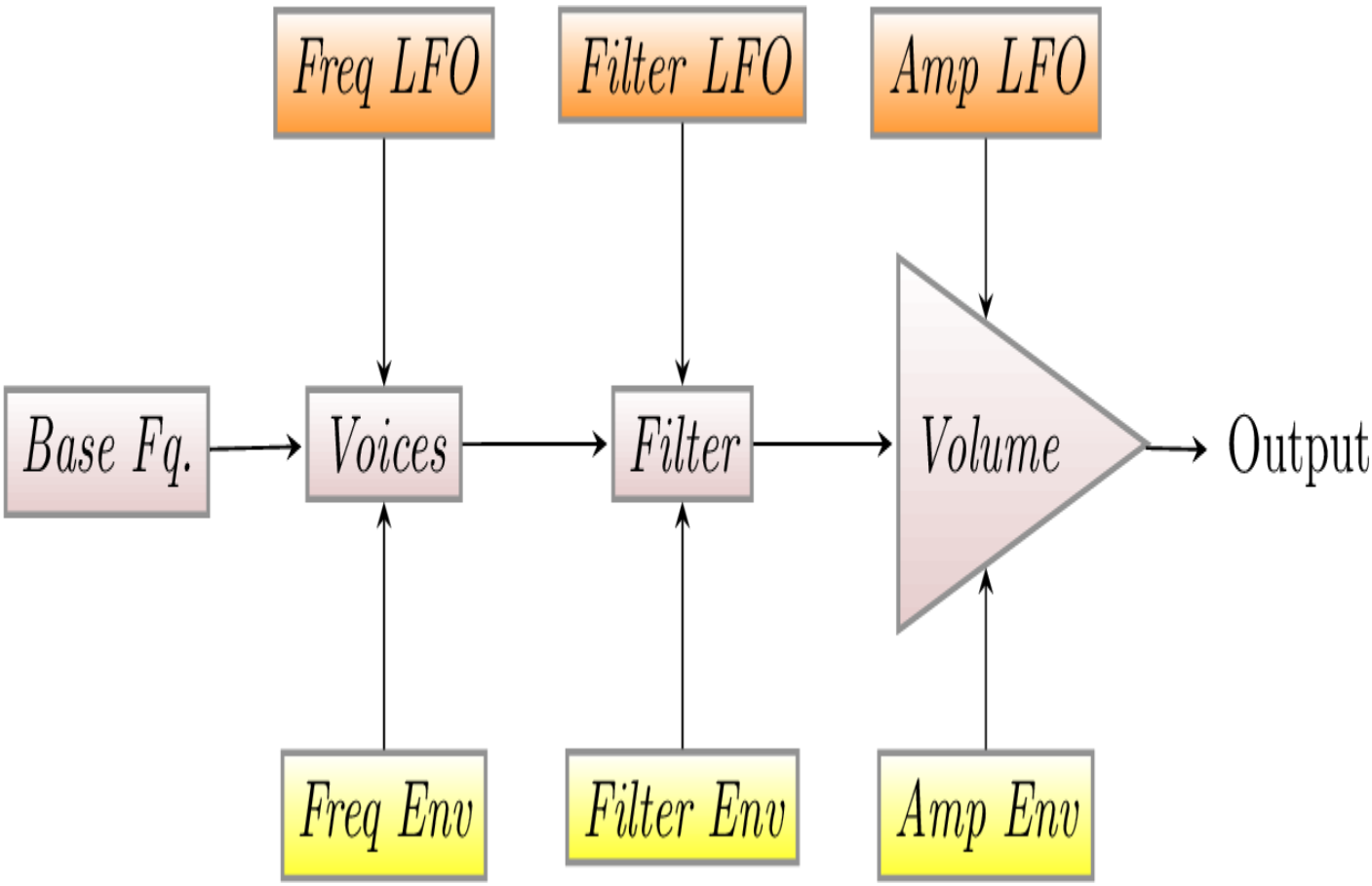


Figure 2. AdSynth Global Elements

The global level of adsynth is almost entirely composed of previously discussed elements. However a few new features appear here, this includes velocity sensing, punch, detune options and realative bandwidth , and resonance.

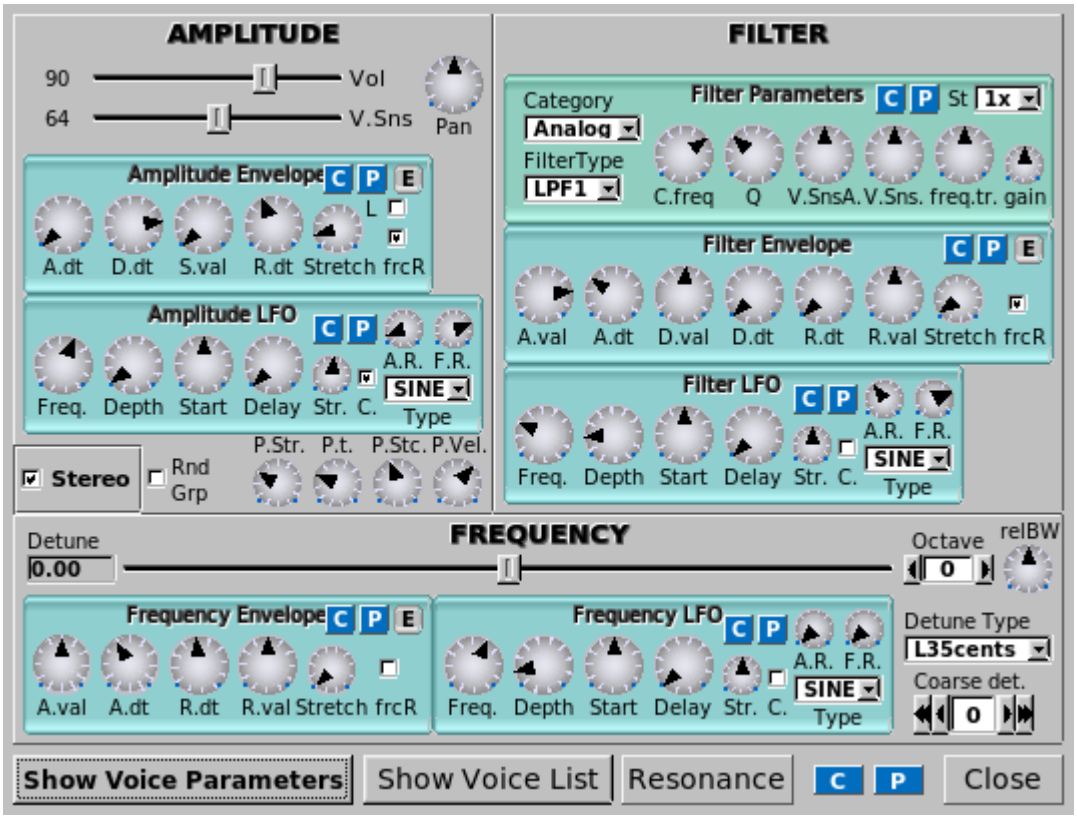


Figure 3. AdSynth Global Window

Velocity sensing is simply an exponental transformation from the note's velocity to some parameter change. The below diagram shows how the velocity senseing controls affects this translation over the whole range of possible note velocities.

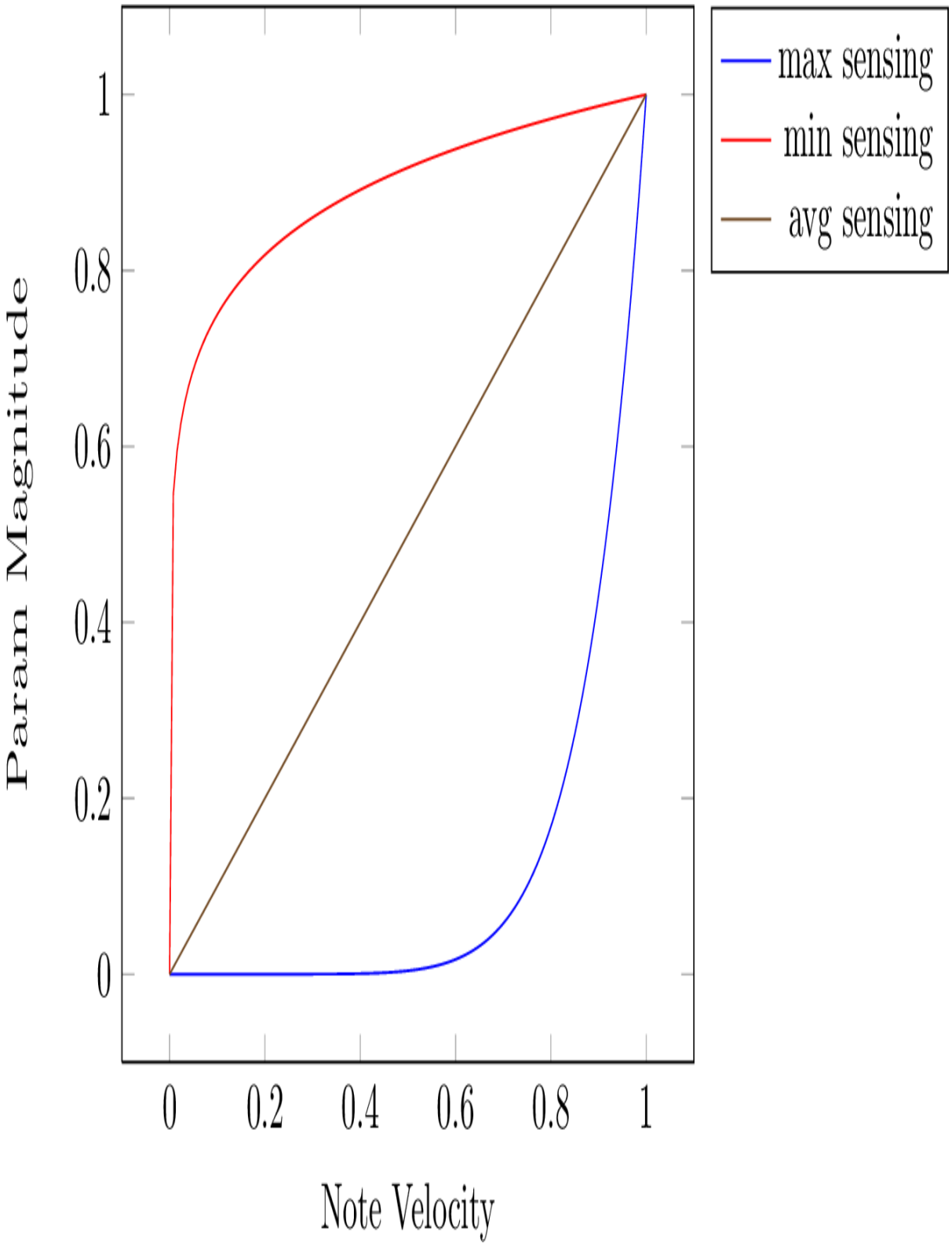


Figure 4. Velocity Sensing Chart

The puch of a note in AdSynth is a constant amplification to the output at the start of the note, with its length determined by the punch time and stretch and the amplitude being determined by the punch strength and velocity sensing. The relBW control in the frequency pane is effectively a multiplier for detuning all voices within an adnote.

> **Note** TODO Talk about resonance

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.

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.

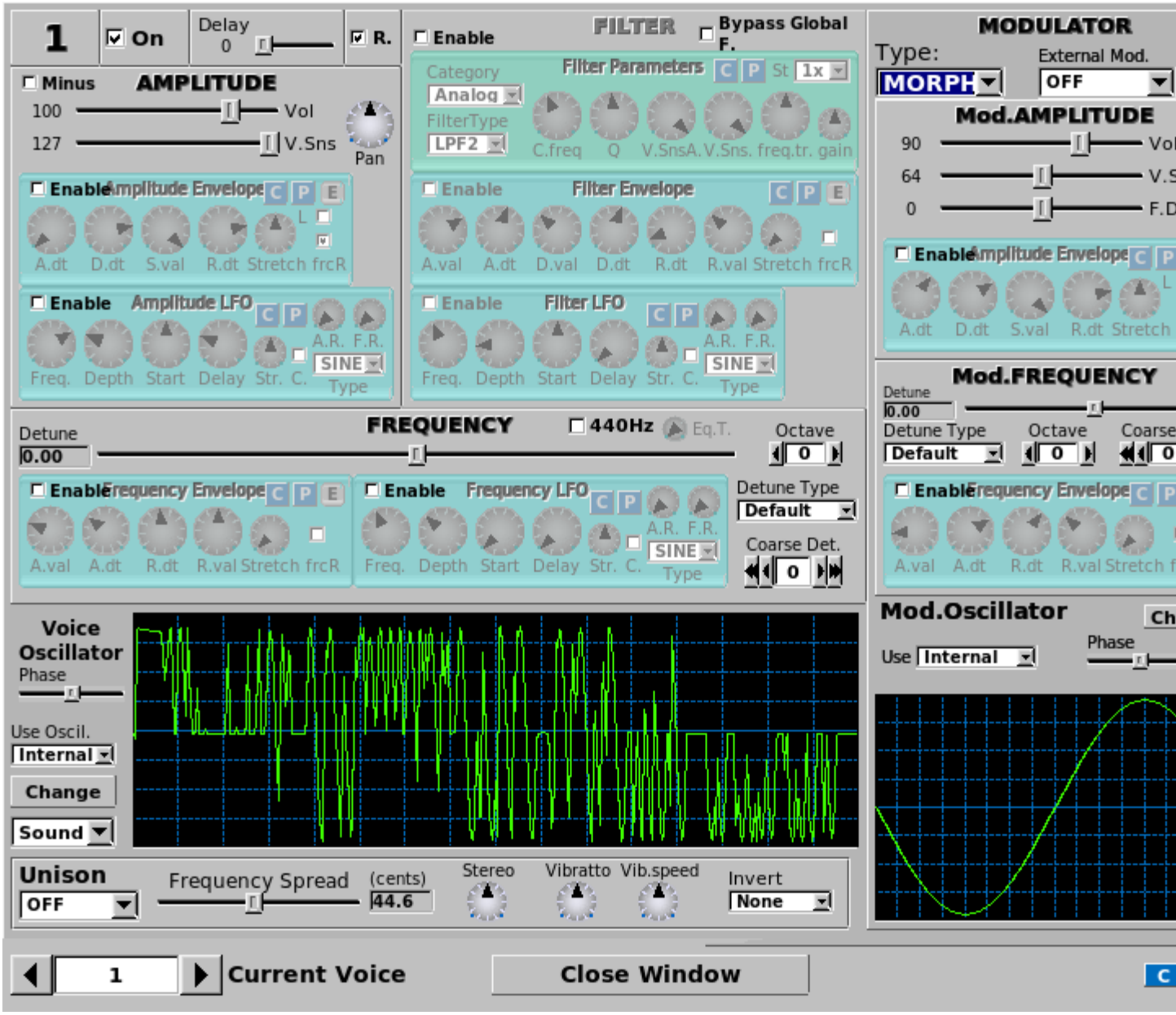


Figure 5. AdSynth Voice Window

5.2.1. Modulation

Within the options for modulation, one can select:

- Morph
- Ring Modulation
- Phase Modulation
- Frequency Modulation
- Disabled

5.2.2. Unison

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

5.3. Oscillator

Note TODO show waveforms, talk about distortions somewhere, etc

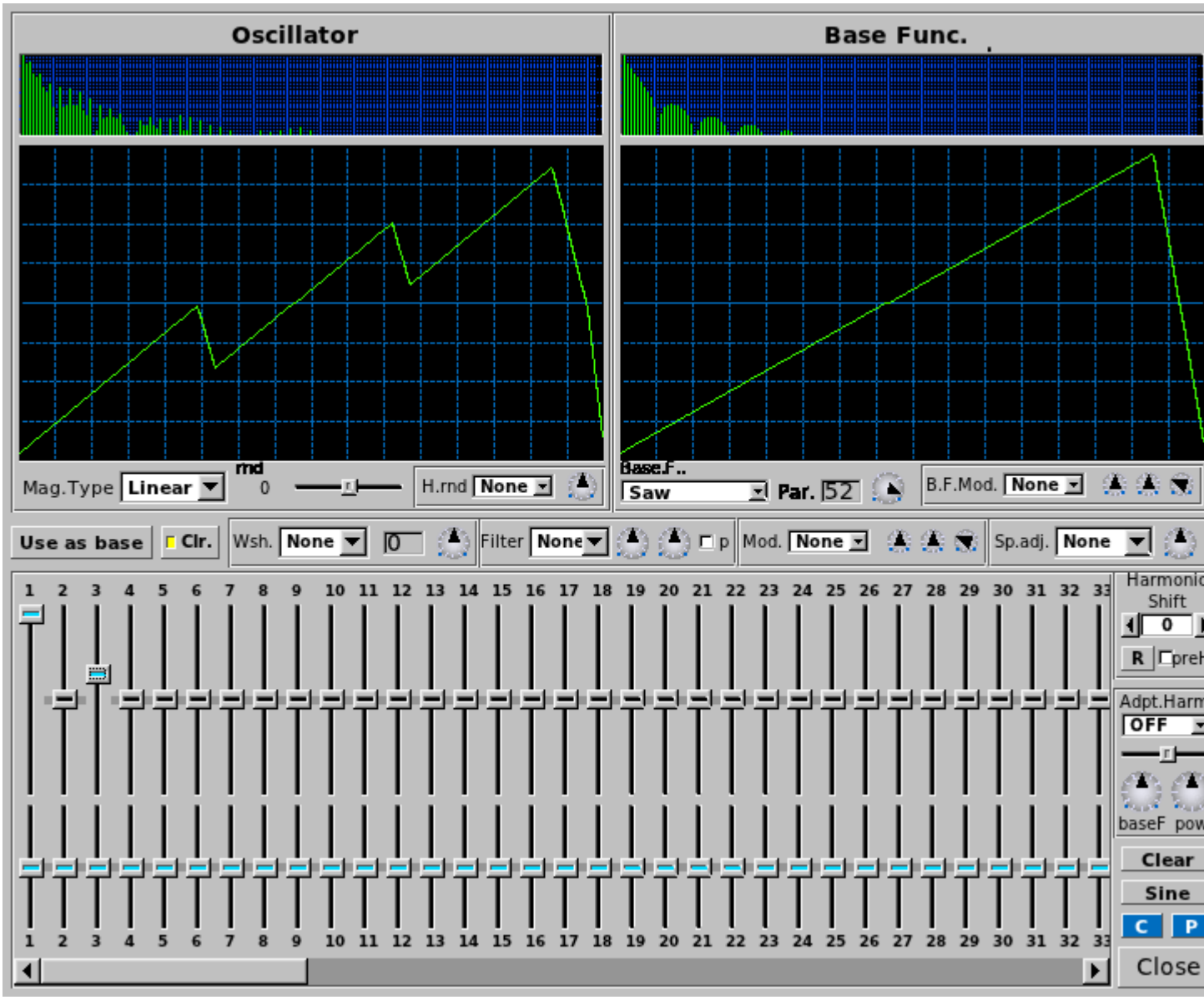
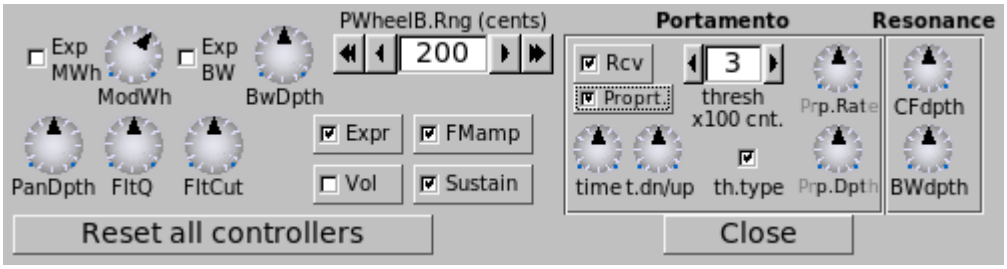


Figure 6. Oscillator Window

6. Controller



6.1. General

- **ModWh**: Modullation Wheel depth
- **Exp MWh**: Exponental Modulation Wheel (changes modulation scale to exponental)
- **BwDpth**: Bandwidth Depth
- **Exp BW**: Exponental Bandwidth (changes badwidth scale to exponental)
- **PanDpth**: Panning Depth
- **FltQ**: Filter Q (ressonance) 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)

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 hundried 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.

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

6.2.1. Proportinal Portamento

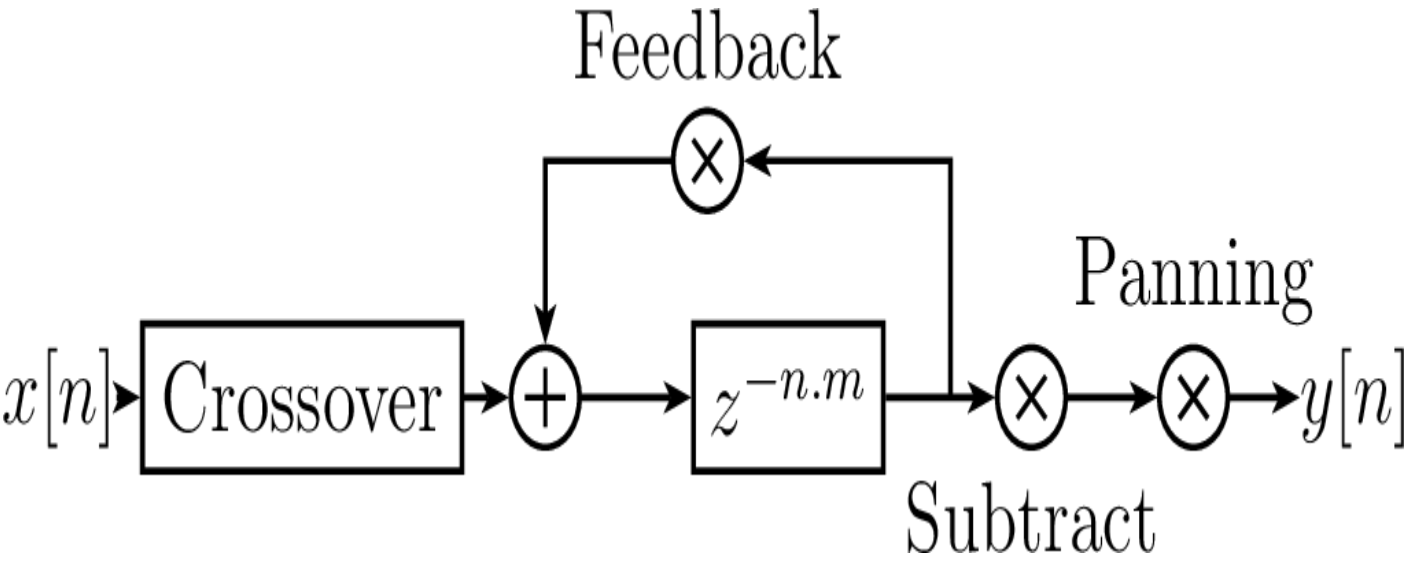
- **Propt.**: If the portamento is proportinal to ratio of frequencies
- **Prp. Rate**: Ratio needed to double the time of portamento
- **Prp. Dpth**: The divergence from

6.3. Resonance

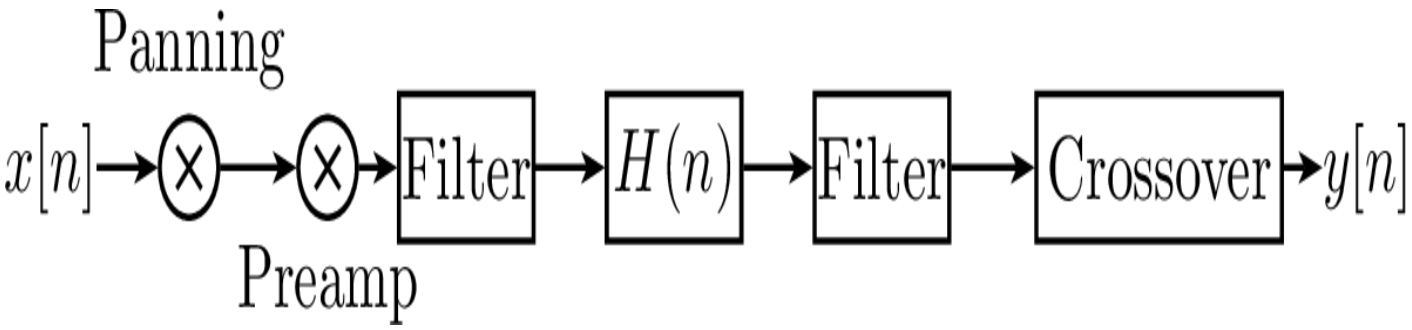
- **CFdpth**: resonance center controller depth
- **BWdpth**: resonance bandwidth controller depth

7. Effects

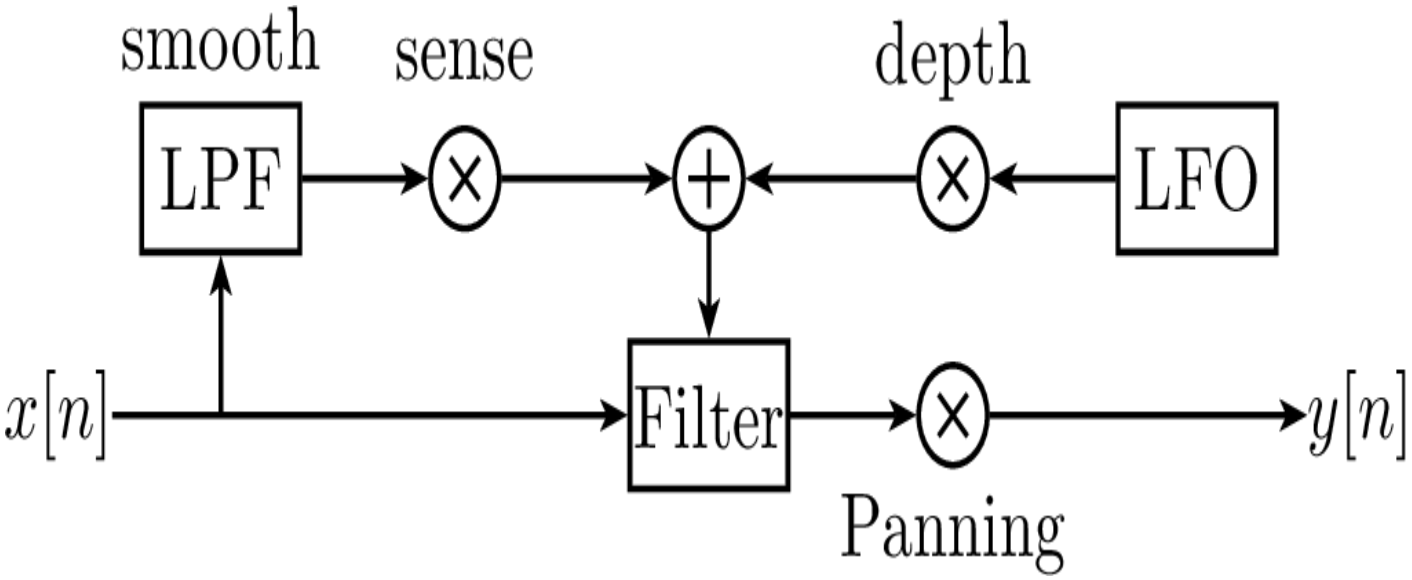
7.1. Chorus



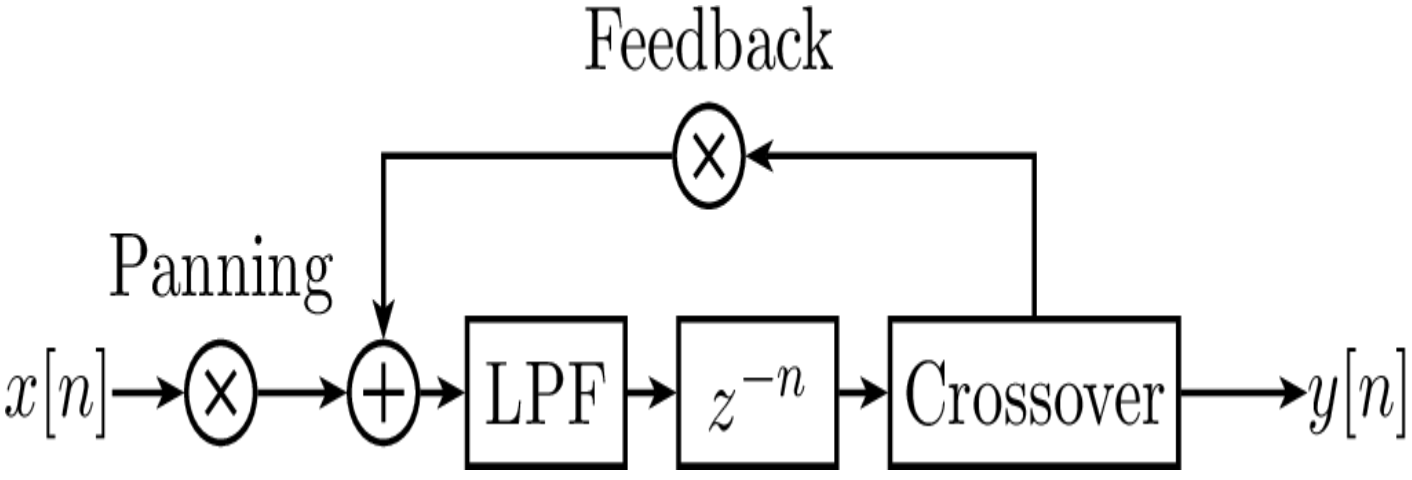
7.2. Distort



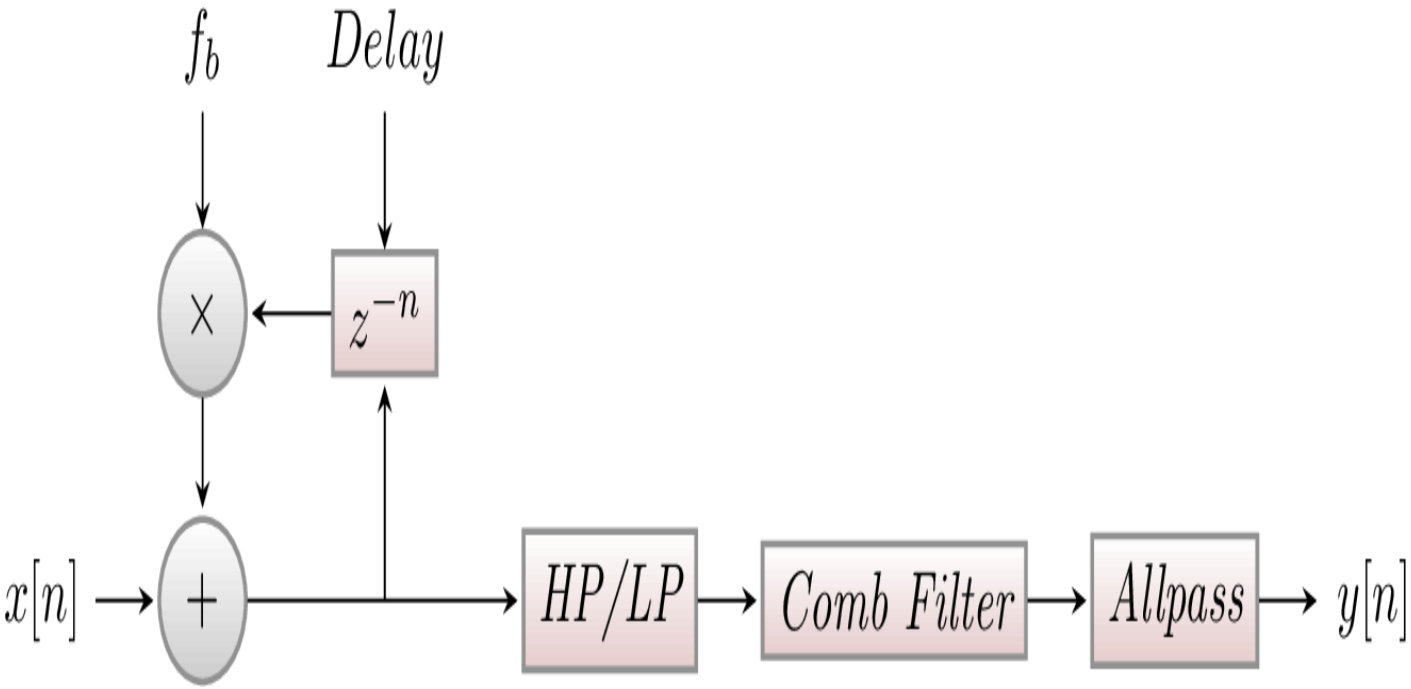
7.3. Dynamic Filter



7.4. Echo



7.5. Reverb



8. Persistence

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

8.1. 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...

8.2. Saving Parts

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

8.2.1. 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.

> **Note** You need to have write permissions to add instruments to the bank.

8.2.2. 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.

8.3. Summary

Extension Summary

```
xmz Everything
xiz Instrument
xsz Scale Settings
xpz Presets
```

9. Appendex 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

10. Appendix B: Building ZynAddSubFX

10.1. Introduction to CMake

Note: This section is mostly copied from the OpenSceneGraph wiki, at:
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 allows you to toggle on/off module compilation and configure various 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.

10.2. 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.

```
#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
```

11. Appendix C: Getting ZynAddSubFX

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

SourceForge

http://sourceforge.net/projects/zynaddsubfx/files/

Distribuition

apt/yum/others

Git

git://zynaddsubfx.git.sourceforge.net/gitroot/zynaddsubfx/zynaddsubfx

11.1. 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 aquanted 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.

11.1.1. Getting the Source Code

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

```
git 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.

11.1.2. 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.

```
#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.

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

---

Last updated 2012-04-18 11:44:53 EDT