

# Xhip

## User Manual

r1196 22 Feb 2020

## Version 8

21 May 2017

# TABLE OF CONTENTS

<b>Introduction</b>	<b>3</b>
<b>Getting started</b>	<b>4</b>
<b>Graphical User Interface</b>	<b>5</b>
User input	5
Display pane	5
Menu	6
LED display elements	7
Logo menu	7
<b>Specification</b>	<b>8</b>
Synthesizer	8
Control	9
<b>Synthesizer</b>	<b>10</b>
Oscillators	11
Mixer	14
Filter	15
Waveshaper	16
Amplifier	17
Envelopes	18
Modulators	19
<b>Control</b>	<b>20</b>
Routing	21
Effects	22
Unison	23
Flags	24
<b>File Formats</b>	<b>25</b>
Xhip	25
Interface	26
PCM (samples)	27
<b>FAQ</b>	<b>28</b>
Preset management	28
GUI updates and skinning	28
Commercialization of Xhip	29
Aliasing mitigation	30
<b>Tips &amp; Tricks</b>	<b>31</b>

# Xhip User Manual

## Introduction

### **Xhip lore**

Xhip was originally designed as a basic synthesizer used inside a [tracker](#) ("acid tracker") around 1999. Before that it had existed in pieces as small tools for generating samples to be used in more common trackers.

In 2003 it existed as a stand-alone application ("acid synth") when someone suggested that it should be converted to a plug-in using a more common and popular interface format. It needed a new name as the synthesizer was originally designed for simple chip-sounds with a little extra and not so much for TB-303 bass-lines or whatever else might be associated with *acid*.

### **Xhip was born**

In 2004 someone suggested it should be released publicly on a web page with news about it posted on [KVR](#). At the time it had no GUI and much more limited capabilities.

How do you say Xhip? Well, the same way you'd say xello, or xlevator, or xurbo xharger of course. Personally I use a sound like rough sand-paper scratching along a plank. Incidentally I wonder if Xhip can produce this sound ... ?

That's enough of a history lesson for now; Let's move on to the...

### **Xweet xubtractive xudio xyntesis!**

With a second oscillator, super-saw oscillator unison, two filters, a wave shaper, pairs of modulators (envelopes and LFOs); Features like ring-mod, x-mod, filter input FM, saturation and global unison Xhip has become far more than its original intention.

Xhip produces high quality dual-oscillator polyphonic synthesizer sounds similar to classic poly synthesizers of the late 80s. Strings, synth pads, basses, leads, organs, pianos, percussion, bells, vocals, sound effects and more are demonstrated in the included bank.

## Getting started

### Init state

The plug-in now uses an *init state* which is the same as the data saved in a project that is loaded with any new instance. This includes the complete state of the plug-in instance; the current bank, MIDI CC mapping, routing, effects, PCM, background color and GUI scaling; all parameters from the *global* section such as polyphony, bender range and glide mode; as well as all the switches such as block DC and mono retrigger.

### Customizing the init state and preset

In order to save the plug-in state you can use the **save state** option in the menu. To save the init state however you need to hold shift while clicking the menu which will add several infrequently used options such as **save init preset** and **save init state**. Once you've customized and saved the init state and init preset you can reload the plug-in in a fresh instance and it will load with your new settings.

All initial configuration will be saved with and over-ridden by any project you load so when you save a project you don't need to worry about conflicting with those projects if you change your init state configuration in the future.

### Factory preset banks

The plug-in does not currently install with any preset banks or other content. You may want to download and load the *Xhip factory bank* and *Synth drums 1* and perhaps save one as part of your customized *init state*. Due to the fact PCM is saved with a state it means an init bank can include sample content.

# Graphical User Interface

## User input

The GUI widgets include text entry boxes, knobs, slide switches, toggles and menus. User input includes **left**, **middle** and **right** mouse buttons, mouse **wheel** and the **shift**, **ctrl** and **alt** keys.

Holding the **shift** key and/or the **right** button while adjusting a knob will provide three levels of input resolution. The **middle** button will reset a parameter to its default value. For knobs, **double-left** or **ctrl-left** will also perform a reset.

The mouse **wheel** can be used to adjust parameters in unit steps; with the **shift** key a fractional step. Some parameters such as *amplitude* or *filter frequency* utilize units such as 3-decibels (fractional: decibels) or octaves (fractional: semi-tones.)

## Display pane

The display pane is in the middle at the top of the GUI. It contains a preset browser, text display, main menu and some LED displays.

### Preset browser

The top of the display pane contains a set of buttons and text-edits. A left-click on any of the text displays will open a preset browser menu to allow a new preset to be selected.

### Preset browsing buttons and index

At the top-left of the preset browser there is a decrement button followed by an increment button. These are followed by a preset index display.

### Preset name, bank name and preset category

Rightward and down a row are three text fields. Right-clicking will allow any of this text to be edited. Only US-ASCII is supported and there is no UTF-8 or other wide-char support.

The preset category can organize a bank into categories such as "Lead" or "Bass". Rather than sequentially listing presets numerically, they are grouped in the preset browser by category.

### Text display

The mid portion of the display pane contains the text display. This shows a text readout for the currently edited parameter as it is adjusted on the GUI. It will also display messages associated with errors or actions taken via menus or switches. The text display shows only the result of GUI input and does not report MIDI input or automation activity.

# Xhip User Manual

## Menu

The lower section of the display pane contains the menu followed by LED-based display elements.

The menu contains options to load or save presets, banks, the plug-in state and PCM as well as MIDI CC mapping and MIDI CC learn.

Additional less often used options are displayed if the menu is clicked while **shift** is held on the keyboard.

### Load and save

These options include the ability to load or save presets, banks and the plug-in state. The extended menu also includes options to load or save the *init preset* and *init state* as well as to recall the default "*factory init*" preset.

### Bank clear and rename

The bank can be cleared using the currently selected preset or the custom init preset as well as all presets renamed as in GM melodic or drum banks.

### CC Map

This sub-menu includes a *learn* item which will trigger MIDI learn and print instructions in the text display. Any existing entry in the CC Map can be removed individually or the entire map can be cleared. In addition the CC Map can be saved as a file and loaded from an existing file so that custom mappings can be created for any controllers you own.

### PCM

Here PCM samples can be loaded from RIFF WAVE (.wav) files. 8-bit, 16-bit and 32-bit float formats are supported but there is no 24-bit extended format support. See the chapter on PCM formats for additional information. All loaded PCM content can also be cleared.

# Xhip User Manual

## **LED display elements**

### **Peak meter**

A row of LEDs following the menu contains a peak meter. Each LED in the peak meter represents a 3 decibel step in peak amplitude. The red LED illuminates where the peak signal level at the output is greater than 0 decibels.

### **Voices count and allocation**

The currently active number of voices is displayed numerically here. Sixteen LEDs below the voices count will illuminate individually while each corresponding voice (1 to 16) is active providing a rough impression of the way voices have been allocated by the synthesizer. These LEDs change color depending upon the state of the amplitude envelope A,D,S,R.

### **MIDI**

The activity LED will illuminate briefly upon any MIDI input to the plug-in. The sustain LED will illuminate while MIDI hold/sustain (CC #64) is held.

## **Logo menu**

The logo can be left-clicked to access a menu allowing the background color to be changed.

It can also be right-clicked to select a scaling factor to resize the GUI.

Some prototype options are included in a menu activated by middle-clicking which will allow a custom background or overlay to be loaded or cleared and reset as well as allowing the background color gradient to be disabled or reset. These prototype options may be interesting to anyone wanting to create custom skins for the plug-in; please contact me via email on the Xhip site for any assistance or a copy of the default bitmaps.

## Specification

### Synthesizer

This table outlines the sections/components of the synthesizer and their parameters. All synthesizer parameters are fully adjustable via parameter automation.

Oscillators	Glide ( time, mode [pitch, Hz] [logarithmic, linear] ) Range ( 32, 16, 8, 4, 2 ) Sync ( A => B, B => A, gate ) X-mod mode ( A, B, [pitch, Hz] [direct, derivative] ) Pitch modulation ( envelope [A, B], modulator [A, B] )
A & B	Waveform ( pulse, ramp, triangle, sine, S&H noise, PCM ) Width X-mod depth
A	Width modulation ( envelope [A, B], modulator [A, B] )
B	Keyboard tracking Tuning ( coarse, fine ) Pitch modulation ( envelope [A, B], modulator [A, B] )
Mixer	Oscillator A ( level, invert ) Oscillator B ( level, invert ) Ring-mod level Noise level
Filter	Mode ( [KHN, KHN 2x] [low, band, high, notch] ) Trigger ( spike, reset ) Frequency Q Keyboard tracking F-mod Q-mod Saturation Frequency modulation ( envelope [A, B], modulator [A, B] )
Waveshaper	Mode ( logarithmic, exponential, absolute ) Depth Symmetry Tone
Post-shaper Filter	Mode ( [6 dB, KHN] [low, band, high, notch] ) Frequency Q Keyboard tracking

# Xhip User Manual

Output	Volume Volume modulation ( modulator [A, B] ) Volume envelope ( selectable [A, B, gate] ) Panning Panning modulation ( modulator [A, B] )
Envelopes	Attack, decay, sustain, release Constant attack time enable Keyboard tracking Trigger ( gate, modulator [A, B], reset, delay [A, B], mod & delay [A, B] )
Modulators	Shape ( pulse, ramp, sine, noise, random ) Rate Delay Width Keyboard tracking Bias ( negative, bipolar, positive ) Range ( low, audio, tempo-synced ) Key-sync enable

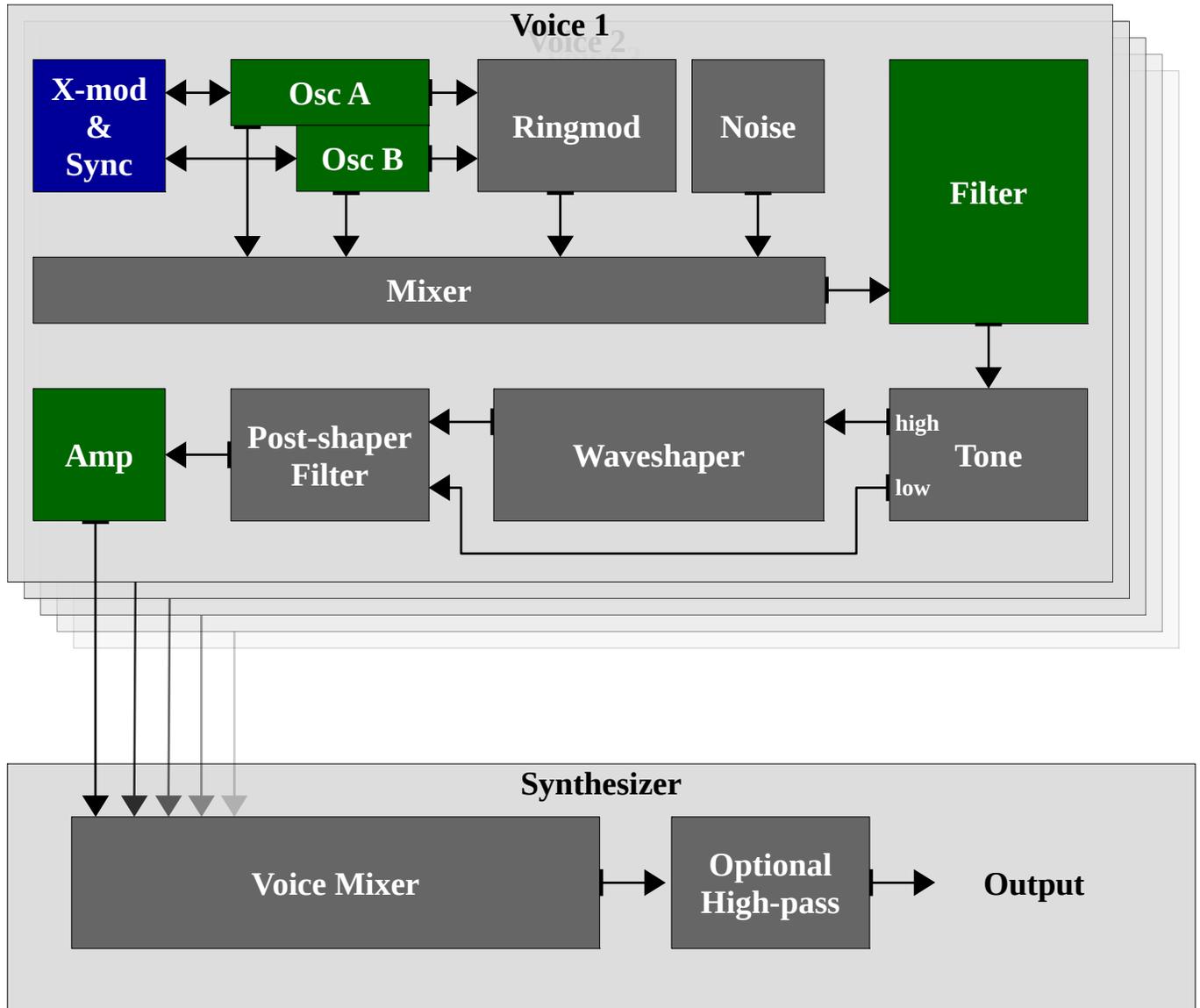
## Control

The control section is not included as part of the core synthesizer parameters stored in presets. Parameters in the control section can not be adjusted via parameter automation.

Global	Tune ( coarse, fine ) Volume Bender range Glide mode Voices
Switches	Use program change, immediate program change, recycle voices, drum mode, mono retrigger, block DC
Unison	Voices, detune, random, width
Routing	Source, mode, destination, depth
Effects	Inserts

# Synthesizer

## Signal Flowchart



### Legend



# Xhip User Manual

## Oscillators

The oscillators section contains parameters affecting both oscillators. Options for sync and X-mod routing as well as pitch modulation including range and glide are available here.

### Glide

The glide mode parameter includes both logarithmic and linear modes applied to either the voice pitch or frequency (Hz) as well as an option to disable the glide thereby removing its processing cost.

In the logarithmic mode a low-pass filter is applied to the pitch or frequency signal. This is the most common glide effect present in synthesizers. In this mode the pitch will change abruptly at first and then more slowly proportionate to the distance from the destination pitch. Due to this property the time to glide between any two semi-tones on the keyboard is equal.

In the linear mode an integrator is applied. This glide effect is seen more often in digital synthesizers and is less common than a logarithmic type. The rate of pitch change is constant in this mode. Due to this property the time to glide between two semi-tones depends upon their distance from one another.

The timing between both logarithmic and linear modes is equal for a glide between semi-tones exactly one octave apart.

A natural consequence of the linear Hz effect is that the rate is proportionate to the position on the keyboard as if key-tracking were applied. The distance between two semi-tones in Hertz at a lower octave is half as much as the next higher octave. Linear Hz mode compensates for this with an automatic adjustment on note-on to ensure the time between any two semi-tones is equal.

### Range

Range or *foot pitch* is a term inherited from pipe organs which represents the transpose in octaves. 8' pitch represents a neutral setting while each halving or doubling represents a transpose upward or downward by one octave.

### Pitch modulation

A mod destination is provided for the pitch shared between the oscillators which may be modulated by envelopes or modulators to create pitch shifting effects such as vibrato.

### Sync (mode)

The oscillators may be configured with sync in either direction. Additionally a **gate** mode is available which will sync both oscillators upon note-on.

# Xhip User Manual

## **X-mod** (mode)

X-mod or *cross modulation* takes the output of one or both of the oscillators and uses it to frequency modulate the oscillators.

Application to **pitch** (exponential) means the target oscillator's tuning will be shifted as depth is increased. It is best used to apply mild audio-rate modulation where the modulator should be anti-aliased. When used in combination with sync the synced oscillator can be forced to stay in tune with the source despite the application of a high level of modulation.

Through-zero **Hz** (linear) modulation means the frequency in Hertz of the target oscillator is modulated directly and can even take on a negative value forcing the oscillator waveform to run backward. This allows equal positive and negative modulation which means the target oscillator remains in tune without its pitch being shifted.

By default these modes sample the input from the source directly. The input can also be read from the **derivative** which means that the modulation value is the difference or change rather than the absolute value. Using the derivative reduces the effect of bias in the input waveform.

When **derivative Hz** mode is used both inherent sources of bias are eliminated which can allow modulation without detuning or "latching up" the oscillator.

The source of bias is the modulating waveform itself. The only waveform which is continuously free of bias is a *pure sine* waveform. Although the **sine** waveform used in Xhip is not *pure* (*a theoretical mathematical abstract*); it is an approximation that eliminates most of the bias applied to pitch. In such a configuration where the source oscillator waveform is unbiased and the modulation is **derivative Hz**, even feedback can be applied without shifting the target oscillator's pitch; to a limit.

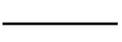
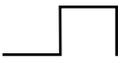
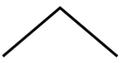
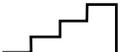
The remaining bias is not due to the method or signals used but rather due to the fact the modulation input is sampled (discrete) at the sample rate ( $f_s$ ) rather than analog (continuous). The complete sum of modulation during the span of time between two *sample points* (instantaneous points in time) is not represented accurately.

It is possible to approximate this sum using integrals of windowed sinc impulses but Xhip does not make an attempt as it is prohibitively expensive to do so. The result is that as feedback and modulation depth are increased beyond a limit determined by  $f_s$ , error will grow and a combination of aliasing, pitch bias and bursts of noise or rarely latch-up of the target oscillators will occur.

# Xhip User Manual

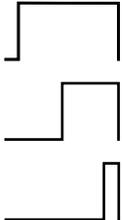
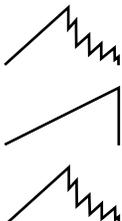
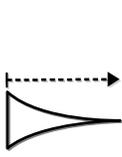
Some parameters are present for both oscillators A and B.

## Waveform

Off		The oscillator may be disabled by selecting off.
Pulse		Standard pulse waveform.
Ramp		Up-ramp often referred to as a <i>saw</i> which may optionally be run in unison.
Triangle		Standard triangle waveform.
Sine		Standard sine waveform.
Staircase		White noise <i>sampled and held</i> at four times the oscillator frequency.

## Width

The width control has a varied use dependant upon which waveform is selected for the oscillator.

Pulse		Used to adjust the duty-cycle of the pulse waveform. At 50% the output will be equal and otherwise known as <i>square</i> and can be adjusted to a narrow pulse at either side of the parameter's range. The continuous width modulation input from envelopes and modulators will take effect <u>only</u> when used with the pulse waveform.
Ramp		Used to enable and adjust the detune amount for the ramp waveform <i>super-saw</i> oscillator unison. With width centered the unison will be disabled and a single ramp will play. As the parameter is adjusted in either direction unison will take effect and detuning will increase.
PCM		Used to apply an offset into the first sample played upon a new note. Although they do not take effect continuously as for the pulse waveform both envelopes and modulators contribute to the offset sampled in the instant when the voice is re-triggered. This allows a noise modulator to be used to set a random offset.

The width control does not take effect for the triangle, sine and staircase waveforms.

## X-Mod (depth)

Controls the amount of X-mod applied to the oscillator. The modulation source signal and method of application are selected in the **Oscillators** section with the **X-mod (mode)** parameter.

# Xhip User Manual

## Oscillator A

Width modulation                      Modulation destination for PWM.

## Oscillator B

Keyboard tracking                      Variable keyboard tracking independent from oscillator A.

Coarse & fine tuning                  Tuning offset.

Pitch modulation                      Modulation destination for tuning offset.

## Mixer

### Oscillator A & B (level, invert)

These parameters control the level of each oscillator. Phase-inversion is also possible using the invert parameter.

### Ring-mod (level)

This parameter controls the level of a ring-modulated combination of the oscillators ( $A \times B$ ).

### Noise (level)

This parameter controls the level of uniform white noise mixed with the other waveforms.

# Xhip User Manual

## Filter

The filter is a single or dual (series) Kerwin-Huelsman-Newcomb filter. This filter type is often referred to as a *state-variable filter* and provides low-pass, band-pass and high-pass outputs.

Both the low-pass and high-pass modes have a slope of 12 dB or 24 dB.

The band-pass mode has a slope of 6 dB or 12 dB. This is one half of the filter slope applied on the low side of the band with one half on the high side.

In place of slope the width or steepness of the notch depends upon the **Q** parameter.

Both single and dual (series) configurations are provided. In the series configuration the second stage **Q** is equal to slightly less than  $\frac{1}{\sqrt{2}}$ . This reduces the amplitude of filter resonance generated by the first stage.

## Trigger

The filter trigger mode can be used to set the filter's *phase* or in other words to input a *spike* (impulse) or to *reset* the filter when note-on occurs. This can be used to immediately begin oscillation at note-on and is useful when producing drum sounds or otherwise where the filter's oscillation is used as a primary component of the sound rather than acting solely upon the input signal from the mixer.

The phase of the band-pass is set to one with the low-pass set to zero. Therefore the band-pass mode will output a *pop* when the trigger is enabled. The low-pass mode will integrate (filter) this *pop*. In the high-pass mode the impulse will be blended between the direct (band-pass) and filtered (low-pass) output according to the **Q** parameter.

## F-Mod, Q-Mod

The F-Mod and Q-Mod parameters control the level of modulation for filter **frequency** and **Q** by the signal input to the filter. That is in other words the signal output from the mixer stage.

## Saturation

This parameter applies non-linearity within the filter which adds harmonic content based upon the input signal as well as filter self-oscillation. In other words the filter becomes sensitive to the signal level. The mixer section includes up to +12 dB of boost in order to make it easy to drive the filter input.

Saturation depth is reduced as cutoff frequency is increased in order to minimize aliasing. Excessive high-frequency and high-gain input may produce significant aliased harmonics.

# Xhip User Manual

## Waveshaper

The waveshaper is a per-voice distortion after the filter and before the second filter. It is limited to a mild range as it does not use any over-sampling or other anti-aliasing.

### Mode

**Logarithmic** applies the sigmoid  $\frac{x}{(|x|+d)}(1+d)$  while **exponential** applies the same sigmoid with a negative coefficient. Exponential mode includes a hard clipping stage to prevent the signal level increasing uncontrollably. **Absolute** mode provides continuously variable half-wave through full-wave rectification.

### Symmetry

Used to apply an offset to the signal before it passes into the waveshaper. This can lead to the production of even harmonic content with increasing asymmetry.

### Tone

The tone control acts as a frequency splitter where the low-frequency content will bypass the distortion.



This allows the waveshaper to be focused on high-frequency content which is useful for adjusting tonal balance or focusing application on filter oscillation or other content without having low-frequency content dominate the sound.

# Xhip User Manual

## **Post-shaper Filter**

The post-shaper filter fills in as both a tone control for the waveshaper as well as for the rest of the synthesizer. It does not provide any modulation inputs although it is capable of keyboard tracking.

## **Mode**

Low-pass and high-pass modes are supplied with both 6 dB RC and 12 dB KHN slopes. Two variations of band-pass and notch modes are provided; a simple minimal RC mode where the **Q** parameter has no effect as well as KHN where **Q** is used to control bandwidth.

## **Amplifier**

The amplifier provides adjustment of output level and panning as well as a mod-destination for each.

## **Amplitude**

The amplitude envelope can be selected from one of envelope **A** , **B** or **gate** which is an attack-release envelope with fixed times. Amplitude may also be modulated by both modulator **A** and **B**.

## **Panning**

Panning may not be modulated by an envelope.

Panning may be modulated by both modulator **A** and **B**.

# Xhip User Manual

## Envelopes

The envelopes are standard ADSR envelopes much like those found in numerous other synthesizers.

### Envelope timing

Timings are displayed based upon the time it takes to travel **99%** of the step size. For example during the release stage of the envelope the peak of the envelope is **1.0** or **0 dB** and the envelope decays toward **0.0** or **-inf dB**. The time displayed is the length of time during which the envelope will have reached within **1%** of the destination. That is **0.01** or **-40 dB**.

This may make the time displayed for envelopes seem unusually high in comparison to other synthesizers which use a measure based upon a smaller segment of the curve.

### Attack stage asymptote

Envelopes are often described referring to the attack stage as *linear*. This is often incorrect. In reality the attack stage is generally the same curve as the decay and release stages however the end point of the curve (called its asymptote) is beyond the peak level of the envelope. In other words the curve is *clipped off* at some point and the decay begins.

If the envelope were allowed to continue to move toward the asymptote it would only reach that point after an infinite amount of time. This is disadvantageous as we likely do not have an infinite amount of time to wait for the decay stage! Common asymptotes are **101%** to **200%** of the peak. This makes the curve appear more linear as varied amounts are *clipped off*.

The envelopes use an asymptote set at **200%** which is slightly more linear than some synthesizers although not entirely linear. The decay stage starts at **50%** between zero and the asymptote.

### Constant attack

Constant attack adjusts the attack time such that regardless of the level the envelope starts from, multiple voices triggered at the same time will always enter the decay stage at the same time.

### Trigger

The envelope is normally triggered by the gate when a note is played. It is also possible to trigger the envelope by combination of the gate and a modulator or to delay the envelope trigger until the modulator's delay time has passed. Another mode allows the envelope to be triggered by a modulator only until the modulator's delay time has elapsed. The envelope may also be forced to reset to zero before the attack stage begins, although this sudden jump will often lead to clicks and pops.

## Modulators

### Shape

Pulse		Standard pulse waveform adjustable between narrow pulses and square.
Ramp		Adjustable between up-ramp, triangle and down-ramp.
Sine		Like ramp with a parabolic shaper applied.
Noise		Normalized noise with a low-pass (red) filter applied.
Random		White noise ramp & hold with ramp time set by the width parameter.

### Width

For the modulators the width parameter takes effect on all waveforms except noise. Pulse can be adjusted between narrow pulses and square. Ramp and sine between up-ramp/s-curve, triangle/sine and down-ramp/s-curve. Random can be adjusted between immediate jumps and linear interpolation.

### Rate

In **low** mode the frequency can be adjusted from the minimum rate at **120 seconds** to the maximum rate at **32.0 Hz**.

In **audio** mode the rate can be adjusted between **-36 semitones** and **+36 semitones** in **10 cent** steps. Tuning is relative to the current note assuming keyboard tracking is set to **100%**.

In **tempo** mode timing can be adjusted between **32 whole triplets** to **64ths**.

### Delay

An exponential attack is applied to abruptly fade-in the modulator. The effect is intended to be quite abrupt and is ideal for delaying the start of a vibrato or other modulation. Delay time can be adjusted between **zero** and **5.5 seconds**.

### Bias

This parameter can be used to scale the modulator waveform by **1/2** allowing it to operate from **-1** to **0 negative** or **0** to **+1 positive** rather than the usual **-1** to **+1 bipolar** range.

## Control

The control section is not a part of the synthesizer itself. Parameters in the control section can not be adjusted via parameter automation. The parameters in the control section are not saved in presets or banks but are saved as part of the plug-in state.

### Glide mode

Depending upon the mode selected for **voices**;

In monophonic modes: **Normal**, **fingered** and **afingered**. In **normal** mode glide is always applied. **Fingered** mode is often also referred to as *legato* where glide is only applied between notes without a release occurring between them. **Afingered** is the opposite in which glide is only applied to notes with a release occurring between them.

In polyphonic modes: **Follow**, **retain** and **voice**. **Follow** sets the start point of the glide for new notes to the most recent note played. **Retain** starts new notes gliding from the current position of the last activated voice; when many notes are played at once they will all glide from the same source frequency. **Voice** mode allows each individual voice to retain its own frequency and upon receiving a new note-on will start to glide from that point.

### Voices

This parameter provides **monophonic** modes including **lowest**, **highest**, **last** and **first** priorities as well as **polyphonic** modes from a **single** voice through to **thirty-two** voice polyphony.

# Xhip User Manual

## Routing

MIDI inputs can be routed to any preset parameter value on the **route** page from the control panel using a table of routings. Such an interface is often incorrectly referred to as a "mod matrix".

Routing MIDI inputs or other sources to preset parameters has the same effect as adjusting each parameter directly via automation, MIDI CC mapping or the GUI with one difference; that polyphonic sources like velocity and poly pressure can be applied to individual voices polyphonically. This system only acts as a part of the controlling interface to the synthesizer voices and does not provide access to internals of a voice such as envelopes, modulators and modulation destinations.

While applying parameter routing and voice recycling at the same time sudden jumps in inputs such as velocity may occur. These changes will be filtered by the parameter filter which is also applied to automation inputs and knobs on the GUI. The result may be undesired; in such a case it is advised that voice recycling should be disabled to avoid the effect. This effect may still occur when a voice is stolen. The only solution to avoid stealing is to increase the total polyphony as the GUI does not provide an option to disable voice stealing.

It is planned that this will be fixed by extending the parameter routing and other functionality in the next version. This functionality is limited by the need for features and space available on the GUI.

### Routing not yet stored in presets

Unfortunately the configuration of the parameter routing is not yet possible to associate with presets. This functionality is planned for the next version or perhaps even an update to the current version and will be included in the alpha version as soon as possible.

This functionality requires extension of the features of the GUI and is restricted by the limited space available there.

# Xhip User Manual

## Effects

Xhip was always intended to support a simple effects plug-in capability which was initially included in the original tracker back in 1999. This version included both a tempo-sync delay and a reverb with send-levels included as preset parameters. Effect parameters were controlled via extended commands in the tracker which would have been unique to each effect type.

In the Xhip plug-in this has been simplified significantly to a simple insert chain of four effects applied immediately after the voice mixer and output of the core synthesizer, output volume level and optional high-pass filter have been applied.

### Effects not yet stored in presets

The configuration of the effects applies to the controlling "main board" of the plug-in and not directly to the synthesizer presets. It is planned that inclusion of the configuration for the effects in presets will be made possible in a future version by extending the functionality of the GUI.

A dynamic implementation is impossible in a multi-timbral-capable synthesizer like Xhip. It will therefore be necessary to allow only manual loading of the effect configuration from presets selected from the GUI as any newly loaded configuration will over-ride the existing configuration.

When operating with multi-timbrality such as where "drum mode" is used, two presets played simultaneously may include entirely distinct configurations. This would not work as there are 128 presets in a bank of which any preset may steal and over-ride the settings of any voice at any time. Due to the memory and processing consumption of the effects it is impractical to allocate and handle an array of 128 configurations and cross-fade send levels between them and up to 512 voices.

While the plug-in is used without multi-timbrality however to browse or play a single preset sound; a single effects configuration is entirely acceptable and will in the future be loaded into the effects section from the currently selected preset when enabled. Such an implementation is one of the features I am most interested in for the next version and will be included in the alpha version as soon as possible.

# Xhip User Manual

## **Unison**

This unison is a *true* voice unison. It activates a number of voices while applying detune to them. These are complete voices including all of the processing such as filters, waveshaper, envelopes and modulators. The CPU time required is exactly equal to that required for an equal number of voices. For example the voice counter will display 64 voices when using 16 voice polyphony and 4 voice unison.

## **Voices**

The number of voices to use for unison from 1 (disabled) to 16.

## **Detune**

The amount of detune to apply between voices.

## **Random**

The depth of random tuning variation applied to each voice. Creates a pitch *humanization* effect.

## **Width**

The width of panning spread automatically applied to unison voices.

This panning is applied in addition to any use of panning by the preset played for a voice and so wide panning of presets is not advised for use in combination with unison as a far-left preset spread far-right by unison will produce a mix level of zero.

Preset panning can still be useful however especially in multi-timbral mode as it can apply slight bias to the resulting panning of each preset. Please keep note that resulting mix levels may be slightly reduced in such a configuration.

# Xhip User Manual

## Flags

### Use program-change

Accept program change messages via MIDI.

### Immediate program-change

Immediately apply parameter changes to active voices upon receiving a program change message via MIDI, the host interface or user input via the GUI. If this option is not selected a voice will retain its settings until it is re-triggered or otherwise reactivated via voice stealing.

### Recycle voices

Recycle the same voice to play any new note matching an active voice with the same note.

This will mean that playing the same note repeatedly with a long release will only use a single voice much like in the case of a piano or other instrument in which each key is directly in control of a single voice rather than being dynamically allocated to a limited number of voices.

It is important to note that with parameter routing and voice recycling used in combination, sudden jumps in inputs such as velocity may occur. These changes are filtered by the parameter filter used on automation inputs and the GUI and the result may be undesired. Voice recycling should be disabled in such a case to avoid the effect. This effect may also occur when voices are stolen. Increasing the total polyphony to reduce voice stealing is the only solution possible in the current version.

### Drum mode

Drum mode allows the synthesizer to operate as multi-timbral such that it can produce multiple sounds at the same time from a single instance. It is most useful for percussion although quite limited currently as it does not support layering or related features and the key map can not be customized.

Notes will be mapped to presets as they are in General MIDI. In order to ease use of this mode an option has been included on the extended main menu (hold **shift**) under "bank: rename" to rename all presets in the current bank as they appear in the GM2 percussive map.

### Mono re-trigger

Re-trigger the voice when pitch changes due to note-on in monophonic mode. This parameter has no effect in polyphonic mode.

### High-pass

Enable a high-pass filter applied to the output of the synthesizer to eliminate sub-sonic content or *DC offset*.

## File Formats

### Xhip

Xhip implements custom file formats rather than depending upon host interface functionality. Part of the reason for this is that Xhip and its file formats existed long before it was wrapped in an external and proprietary interface. It was designed with its own interface without any expectation that it would later be wrapped in one providing any such functionality.

### Preset (.xhippreset)

Xhip presets contain the value of all of the main synthesizer parameters. Additional *global* settings from the control section and other data such as PCM are not stored in Xhip format presets.

### Bank (.xhipbank)

Xhip stores all of the 128 *user bank* presets in its own bank format. Additional *global* settings from the control section and other data such as PCM are not stored in Xhip format banks.

### State (.xhipstate)

Xhip state files store the complete state of a plug-in instance including the complete bank, currently loaded PCM data and routing and effects configurations all the way through to the currently selected background color and GUI scaling factor.

# Xhip User Manual

## Interface

The host interface provides functionality via parameter and *data chunk* get and set methods which allows the host to store both presets and banks as well as complete state information for project files.

The plug-in depends upon the host implementation for support of interface format files via the host's interface implementation and does not support such file formats internally.

### Preset (.fxp)

This functionality depends upon the host implementation although in most cases .fxp format files contain the preset chunk provided by the plug-in. Xhip encapsulates the current preset in .xhippreset format in the preset chunk provided to the host.

### Bank (.fxb)

Unfortunately the host interface does not distinguish between banks (a collection of presets) and the plug-in state saved in projects. Generally plug-ins will respond to a host request to save the bank chunk with the complete plug-in state. This allows the host to reload the complete state with project files.

Xhip includes all of the content as it would be saved in the internal .xhipstate format in the data chunk provided to the host.

# Xhip User Manual

## PCM (samples)

Xhip supports only 8-bit or 16-bit integer or 32-bit floating point RIFF WAVE PCM format files. Multi-channel files are supported although only the first channel will be loaded with the further channels ignored.

PCM data can be loaded and used to replace the wave shapes built in to the oscillators. The data is output by using nearest-neighbour interpolation (zero-order hold) with the same anti-aliasing filter as used on the other waveforms.

Unfortunately PCM is not saved in presets or banks although it is saved in projects with the plug-in state (xhipstate, fxb.) Without additional supporting elements on the GUI the ability to configure PCM functionality is limited within the plug-in.

The PCM functionality can be used to significantly extend the range of oscillator timbres which can be produced. This includes playback of fully anti-aliased sample content including for example drum sounds or vocals.

## RIFF WAVE chunks

The RIFF WAVE format includes special extensible chunks which can contain additional data such as sample names, loop options, tuning and other parameters. This data is possible to add using an external *wave editor* tool and will be loaded by Xhip.

Chunks and data supported by Xhip include:

LIST.INFO.INAM	Sample name.
smpl	Tuning offset and loop points.

## Xhip .wavs file list format

Xhip also supports a custom text-based sample list format which can be used to speed up loading of multiple frequently used files. The file is a simple plain text file with the path to each sample that should be loaded on its own line followed by a new-line.

Blank lines are ignored but leading and trailing whitespace is not ignored. The code is Windows specific and requires that all paths utilize back-slash characters and point to a valid RIFF WAVE file.

Since PCM storage in the state chunk and projects was implemented this file format has become less useful. It is not recommended that this method should be used in the general case.

## FAQ

### Preset management

*When I change some settings on a preset the changes will persist if I move to another preset and back. The only way I can get back to the original unmodified preset is by loading the original preset from a file. Is this is a bug?*

Not at all.

The advantage is that it isn't necessary to remember to *save* a preset as this occurs automatically as parameters are edited. If an *edit* or *working state* were used the preset would need to be saved manually to the *user bank* in order to ensure changes were maintained.

This would make complete control including parameter adjustments and program changes via an external MIDI controller impossible as there would be no way to save parameter adjustments without opening the GUI. Unfortunately MIDI does not include preset management functions such as save, load or init.

Xhip shuffles these issues around ever so slightly to make itself controllable via MIDI.

If a *permanent* save of a preset or bank is needed it must be manually saved to a file. Otherwise the entire *user bank* (in memory, per instance) is used as a *working state* and there should be no need to worry about forgetting to save changes before switching presets.

### GUI updates and skinning

*Are there any plans to update the Xhip GUI? What about skinning support?*

Xhip includes some rudimentary skinning support in order to support the existing GUI. All the GUI resources can be made available to anyone interested in working on new artwork or layouts and such interest will provide motivation to improve the skinning system to support any new features required. The GUI includes bitmap resources such as backgrounds, knob strips, switch strips, LEDs, fonts, coordinate tables for the layout of controls on the GUI as well as menu layout tables.

I do not consider myself an artist. Although I do invest reasonable effort toward improving my skills I am certainly not talented when it comes to producing aesthetic works of art. Therefore an artist would need to be hired to produce a layout and graphical elements of a quality that would satisfy the aesthetic appetites of users. Skinning support will be improved over time which will make such aesthetic projects more practical in the future. One might hope then that art assets and skins are created and made available.

# Xhip User Manual

## **Commercialization of Xhip**

Unfortunately there is little reason to invest further effort toward Xhip for the purposes of commercial exploitation. The number of users and interest in the plug-in has always been very small such that even given that if hypothetically 100% of those users were to pay some significant amount (\$100?) for the plug-in, it would not justify much additional effort. This topic is often brought up by individuals seeking improvements to the plug-in which they feel would be made possible in some way if the plug-in were to be exploited commercially.

The reality is in fact the opposite. Such exploitation can only take place given the presence of such features aimed at creating a saleable product. Often it is suggested that Xhip would sell well in its current condition. In reality it does not attract much attention even as a free plug-in which does not provide sufficient evidence to support such an assertion.

Xhip will continue to do and focus on the things that I found severely lacking in commercial plug-ins. Many of these features are not often sought after by a majority of plug-in customers and so those with a need for such features are often ignored by the market due to lack of a profit motive. Sometimes when you can't otherwise get a job done right, you have to do it yourself!

# Xhip User Manual

## Aliasing mitigation

In most cases where aliasing is an issue the only true solution is to apply different techniques which generate reduced or zero harmonic content. Where this is impractical the only remaining mitigation is generally some form of over-sampling.

Most presets do not suffer from aliasing, yielding a very poor cost vs. benefit for mitigation techniques.

In Xhip common sources of aliasing are:

Audio-rate modulation    Audio-rate modulators, X-mod and filter input modulation of frequency or Q.

Non-linear processing    Ring-mod, filter saturation and the waveshaper.

Clipping                    High signal levels; oscillator signals and filters at the same frequencies causing constructive interference that builds beyond available headroom.

During over-sampling the input signal must be interpolated to produce a band-limited version of the input. If the process involves Nth order non-linearity it requires minimally a N-times (**Nx**) over-sample. This signal is then processed at the elevated rate and the harmonics then fit within the larger band. As a final step the signal is filtered again to remove any content outside the original band before **Nx-1** of the samples are thrown out. Over-sampling requires at minimum **Nx** as much processing plus the overhead of any interpolating filters used. Over-sampling is a very complicated, computationally intensive task.

Due to the complexities of multiple possible sources of aliasing, extreme variability in results, latency introduced and computational cost it is usually not sensible to apply such techniques internally. For audio-rate amplitude modulation **2x**, filter saturation anywhere between **2x** to **5x** may be satisfactory. Cases such as X-mod, clipping or the waveshaper may require **20x** or more to eliminate audible aliasing. The resulting timbre would differ depending upon exactly how the technique was applied.

Handling this directly would be incredibly complex and make Xhip more an *oversampler* vs. *synthesizer*. It is best in all but the most simple, clear-cut cases to leave it to the user to decide when and how to act to achieve their desired results. Rather than complicating Xhip, the plug-in can be rendered at any sample rate and processed at vastly higher quality, more flexibly, *offline*. While such a full over-sampling technique has additional overhead the results are often favorable and tend to produce a more *natural* timbre in the simplest way possible.

## Tips & Tricks

These are some very basic techniques that can be used to achieve things that may seem initially not possible in Xhip. With any well-functioning synthesizer there are many alternative methods to achieve different effects that are often not initially obvious without one "opening one's mind". This is part of what makes synthesizers very interesting instruments not only in a musical sense but also in terms of exploration and expression through technique and sound alone rather than solely when combined with melody and rhythm.

### Post-shaper filter

The post-shaper filter can be used as a very simple EQ in either low, high or band/notch modes to pick out specific frequencies. While using the 12 dB high-pass mode tuned to ~52 Hz the Q parameter can be adjusted to boost bass frequencies.

Another common application is to add peaks or overtones using the 12 dB modes with Q near 100%. The primary sound may be generated by the oscillators and main filter with a bell-like overtone added by the second filter acting on harmonics added by the waveshaper.

### X-mod to pitch with sync

An interesting effect can be achieved by applying sync **A** → **B** and X-mod source **A (pitch)**.

This means oscillator **A** is used as the source for both X-mod and sync and applied to the pitch of the oscillators. With this setting the X-mod depth parameter for oscillator A acts as feedback while the parameter for oscillator B acts as modulation depth.

Normally application of X-mod to pitch would detune the pitch of the destination oscillator. By applying sync at the same time the target oscillator can be forced to operate at a fraction of the sync source's frequency regardless of the amount of pitch modulation applied. The pitch of the sync source must remain unmodulated by X-mod to ensure it remains free of pitch bias which means the feedback modulation depth for the source oscillator must remain zero.

### FM sounds with X-mod depth by MIDI CC

Xhip in its current form lacks a user accessible internal modulation and signal routing system ("*mod-matrix*"). It does implement parameter routing external to the synthesizer which can route input parameters like polyphonic key pressure to any of the externally accessible synthesis parameters.

This means it is possible to configure polyphonic pressure with oscillator X-mod depth as a destination. It is also possible to use external MIDI CC generators to produce an envelope, LFO or other modulation signal polyphonically.