Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Overview 機能概要

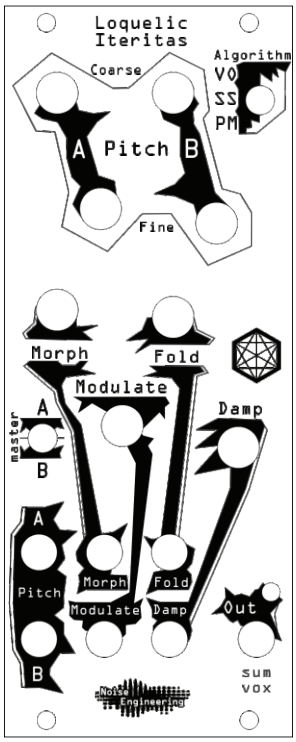
タイプ	VCO
サイズ	10hp
奥行き	26mm
接続	2x8 Eurorack
+12V	150 / 80mA(5V使用時)
-12V	5 / 5mA
+5mA	0 / 90mA

“これは人を殺しかねない。”
-- DJ Surgeon

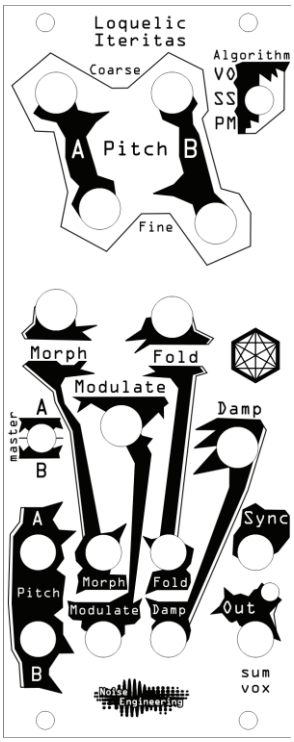
“聴いたこともない膨大なイカしたサウンドの宝庫だ。”
-- Surachai

Loquelic Iteritasはデュアル・ピッチ・コントロールを内包する3種類の伝統的なシンセジス・アルゴリズムに基づくデジタルVCOです。
4種類の音質形成と2種類のピッチ・コントロールによってサウンドのパラメーターを決定して膨大なヴァラエティに富んだサウンドを生成します。

シリアル・ナンバー555番以前



シリアル・ナンバー555番以降



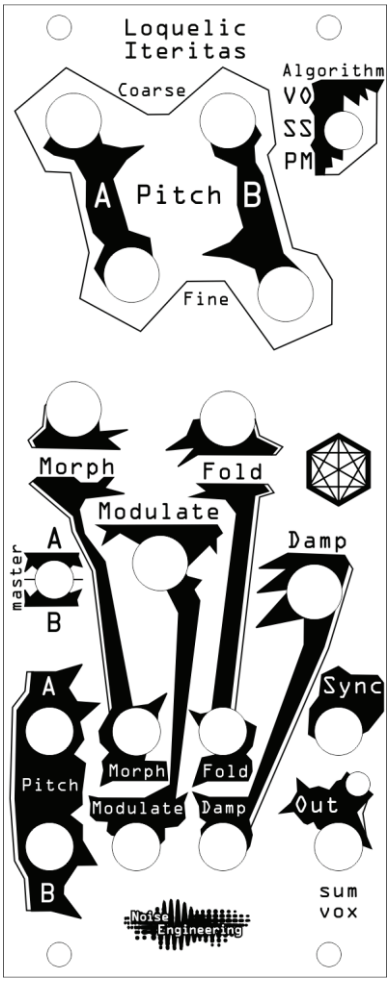
Noise Engineering

Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Patch Tutorial パッチ・チュートリアル



Loquelic Iteritasを知るために最も簡単な方法は各ノブを回しながら
そのサウンドを聴くことです。出力をミキサーへ繋ぎ、早速弄んでみましょう。

Loquelic Iteritasは連続的な音質形成コントロールを持ち味とします。
お好みのLFOを4種類の音質コントロール入力へ送ってみましょう。
(Morph,Fold,Modulate,Damp)

また2種類のピッチにそれぞれ独立したコントロールを行うことでおもしろい効果を
生むことができます。
(デフォルトでは1V/OCT入力は互いのピッチ・コントロールへノーマライズ/内部接続されます。)
特に弊社別売りのモジュールであるTonnetz Sequentを用いて音程を作ることで
興味深い結果が得られるでしょう。

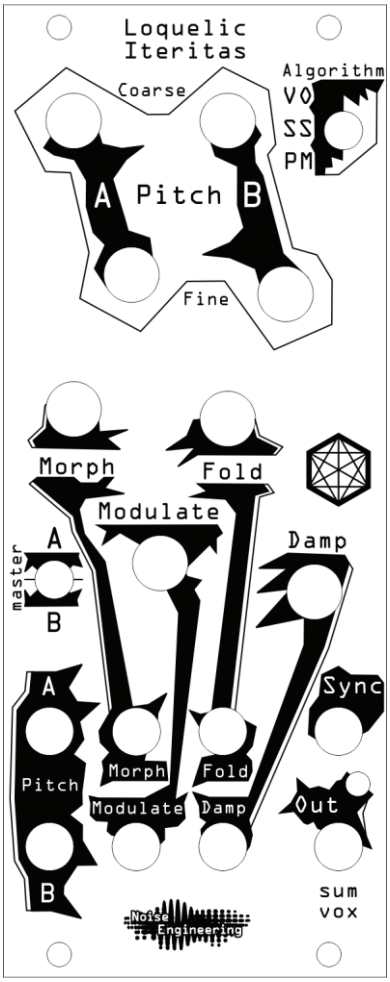
Noise Engineering

Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Interface インターフェース



Pitch A

オシレーターAのピッチは1v/Oct入力によってコントロールされ、FineとCoarseノブによって設定されたオフセット値が入力値へ加算されます。Pitchインの片方に入力されたCVはもう一方へ自動的にノーマライズ(内部接続)されます。

Pitch B

オシレーターBのピッチは1v/Oct入力によってコントロールされ、FineとCoarseノブによって設定されたオフセット値が入力値へ加算されます。Pitchインの片方に入力されたCVはもう一方へ自動的にノーマライズ(内部接続)されます。

Damp

音質コントロールです。以降のページで各パラメーター毎に異なるこのパラメーターの働きを詳しく説明致しましょう。

Mod

音質コントロールです。全モードにおいて2つのピッチ・オシレーター間のフェース・モジュレーション(位相変調)をコントロールします。

Fold

音質コントロールです。全モードにおいて波形フォールド回路のスレッショルド値をコントロールします。

Morph

音質コントロールです。全モードにおいてオシレーターの波形をサイン波から三角波、ノコギリ波へ連続的に変容させるコントロールとして働きます。

Algorithm

アルゴリズムの選択に使用します。詳細は以降のページで詳しく説明致します。

Master

内蔵の2つのオシレーターのシンクをコントロールします。センターの位置で両オシレーターは完全に独立して稼働します。Aが選択されるとオシレーターBがオシレーターAへシンクします。同様にBが選択されるとAがBへシンクします。

Sync

Syncへの入力シグナルは立ち上がり毎に内蔵オシレーターの状態をリセットさせます。いわゆるハードシンクに使用してください。この入力はシリアル・ナンバー700以降に配備されています。

Out

ACカプリング・オーディオ出力です

Noise Engineering

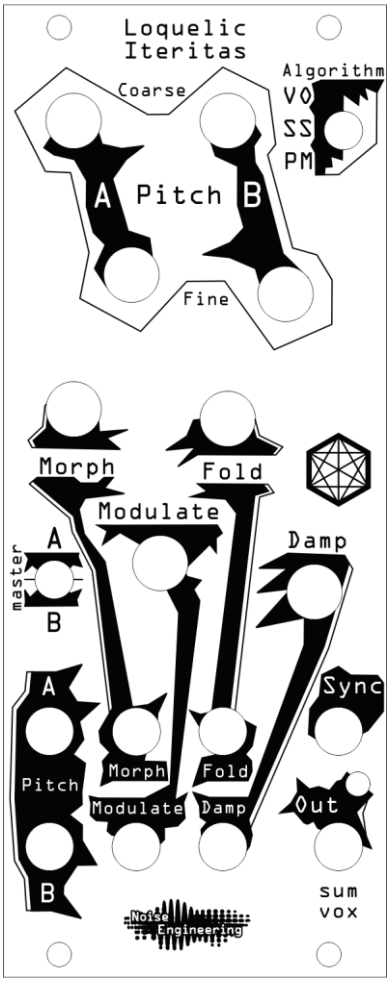
Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

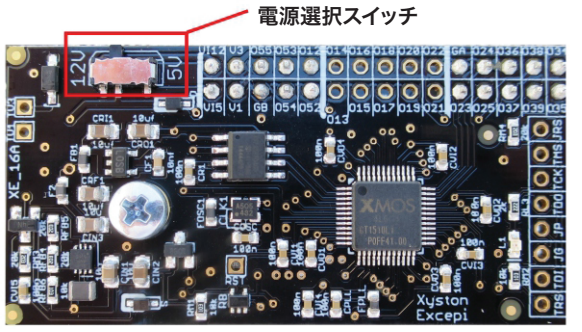
Calibration キャリブレーション

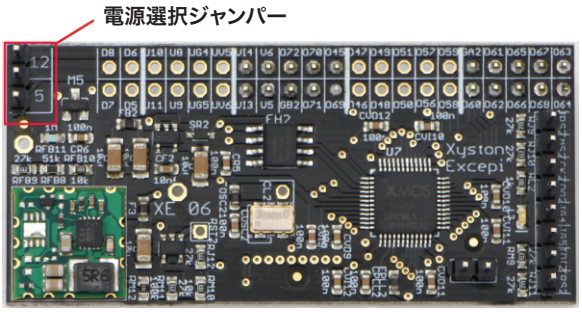
Loquelic Iteritasはストロボスコープを用いてピッチ・レンジ間のオクターヴ・チューニングをキャリブレーションするのが最も適しています。各ピッチ入力は独立したキャリブレーションを配備しています。各ピッチは単一入力で両方の基本ピッチを定められるよう、マスター・スイッチを使用することで内部接続の有無を切り替えすることができます。

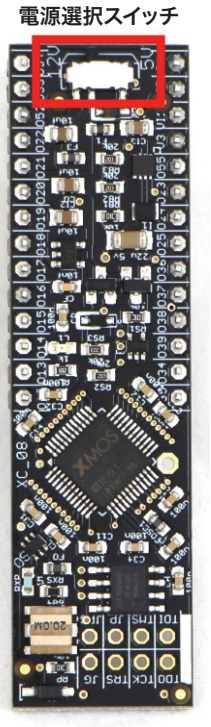


Voltage Supply 電源規格

Loquelic Iteritasは5Vユーロラック電源レールを使用することで12Vバス内のノイズを削減させます。このモジュールはこれまで3種類の異なるCPUボードで生産されており、内2種類がスイッチで、1種類がジャンパーでお使いのシステムに合わせて使用電源レールを切り替えることができます。スイッチ式CPUの場合は使用される電源レール側に丁寧に切り替えてください。ジャンパー式CPUの場合は中央のピンからジャンパーを取り、使用される電源レールが印字されたエリアにジャンパーをインストールしてください。





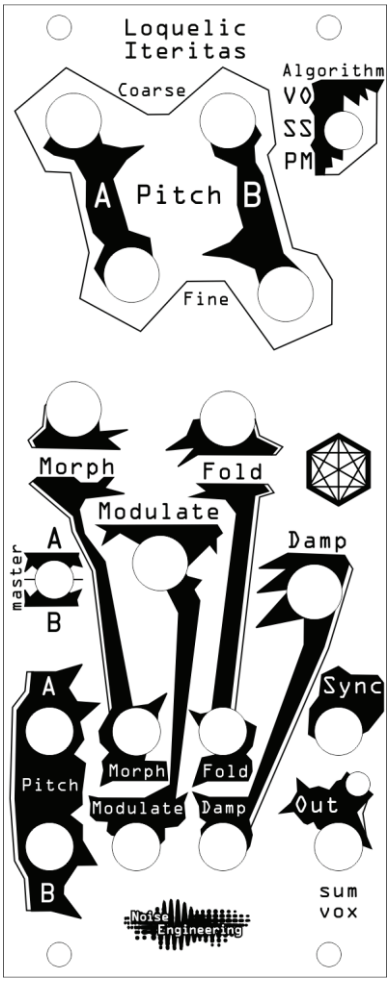


Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Algorithm: V0 アルゴリズム: VO



VOアルゴリズムはカーティス・ローズの著書である”マイクロサウンズ”で論ぜられている
VOSIMアルゴリズムをおおよそ基にプログラムされています。
このアルゴリズムはキャリアをエクスポネンシャル(指数関数)による振幅変調を行うことで
より複雑な倍音構造を生成します。
VOアルゴリズムにおいて最もシンプルなキャリア波形となるシヌゾイドはキャリアが
センターの際にガウス分布を伴ったスペクトラムを生成します。
キャリアを波形をより複雑化させることで各倍音毎にガウス分布を生じさせ、
コム・フィルター・ノイズに似たスペクトラムを生成します。

Pitch Aはキャリアの基本周波数を決定し、Pitch Bは減衰指数の周波数をリトリガーします。

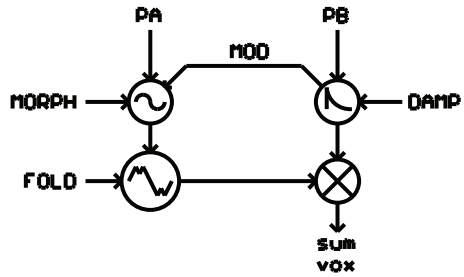
Interface インターフェース

MORPH - オシレーターAの波形を変化させます。
DAMP - オシレーターBの各周期終わりの一定の減衰を設定します。
MOD - オシレーターAをオシレーターBで位相変調します。
FOLD - 最終波形フォルダー回路の波形フォールド・スレッショルド値を設定します。

References 参考文献

Kaegi, Werner, and Stan Tempelaars. "Vosim-a new sound synthesis system."
Journal of the Audio Engineering Society 26.6 (1978): 418-425.
Roads, Curtis. Microsound. MIT press, 2004.

Loquelic Iteritas
Mode: V0



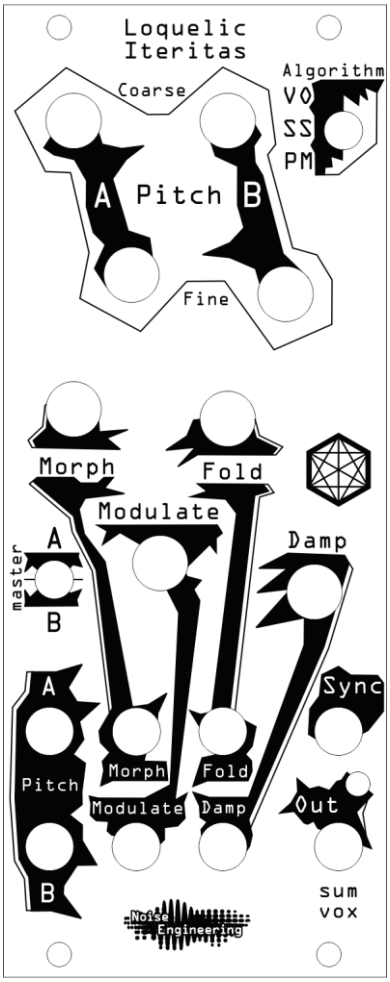
Noise Engineering

Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Algorithm: SS アルゴリズム: SS



SSアルゴリズムはジェームズ・ムーアによって開発された離散総和式シンセシスを
高度に改良したプログラムです。
前提は発散無限級数と比較的容易な計算の間でシンプルな数学的等式で表されます。

方程式:

$$\frac{sin(\Theta) - a\ sin(\Theta - \beta)}{1 + a^2 - 2a\ cos(\beta)} = \sum_{x=0}^{\infty} a^x sin(\theta + x\beta)$$

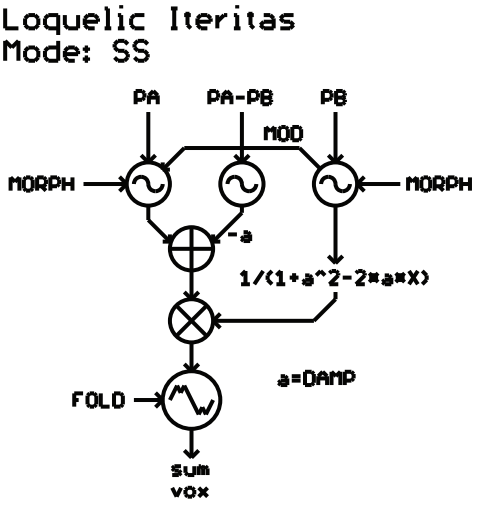
この方程式は様々な幅広い音楽的な響きを持つスペクトルをたった2つのパラメーターで
生成します。Loquelic Iteritasは正弦波の項をマルチ波形型のオシレーターへと汎用化させます。
これらの2機のオシレーターは2つのピッチ入力を追従しながら、2つのピッチ入力の差異を
3つめの追従とし、より多くの倍音を加えるための波形フォルダーをその値でコントロールします。
等式においてオシレーターAは左側に正弦波の項が分子に表され、
オシレーターBは正弦波の項が分母に表されます。

修正方程式:

$$\frac{sin(w_A t) - a\ sin(w_A t - w_B t)}{1 + a^2 - 2a\ cos(w_B t)} = \sum_{x=0}^{\infty} a^x sin(w_A t + x w_B t)$$

Interface インターフェース

MORPH - 全オシレーターの波形を変化させます。
DAMP - 等式におけるAのパラメーター値を設定します。
　　このコントロールを高めることで高倍音域のスペクトルを生成します。
MOD - オシレーターAをオシレーターBで位相変調します。
FOLD - 最終波形フォルダー回路の波形フォールド・スレッショルド値を設定します。



References 参考文献

Moorer, James A. “The synthesis of complex audio spectra by means of discrete summation formulas."
Journal of the Audio Engineering Society 24.9 (1976): 717-727.
Jolley, Leonard Benjamin William, ed. Summation of series.
Courier Corporation, 2012.

Noise Engineering

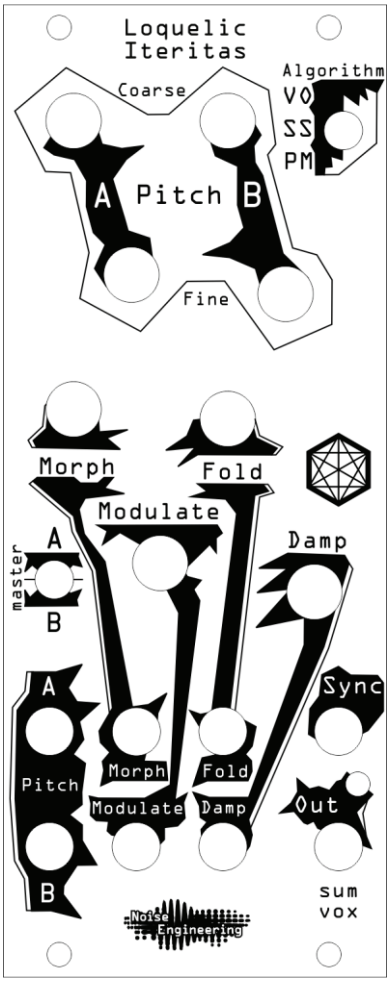
Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

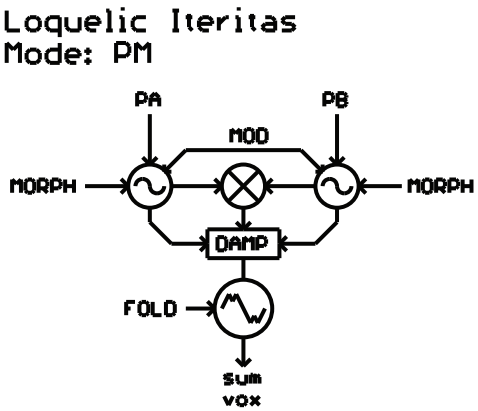
Algorithm: PM アルゴリズム: PM

PMアルゴリズムは単純な2機のタイムドメイン・オシレーターによる位相変調によって両方の振幅変調を組み合わせるものです。



Interface インターフェース

MORPH - 全オシレーターの波形を変化させます。
DAMP - オシレーターAと振幅変調されるBのブレンドを設定します。
MOD - それぞれのオシレーター同士で位相変調をおこないます。
FOLD - 最終波形フォルダー回路の波形フォールド・スレッショルド値を設定します。



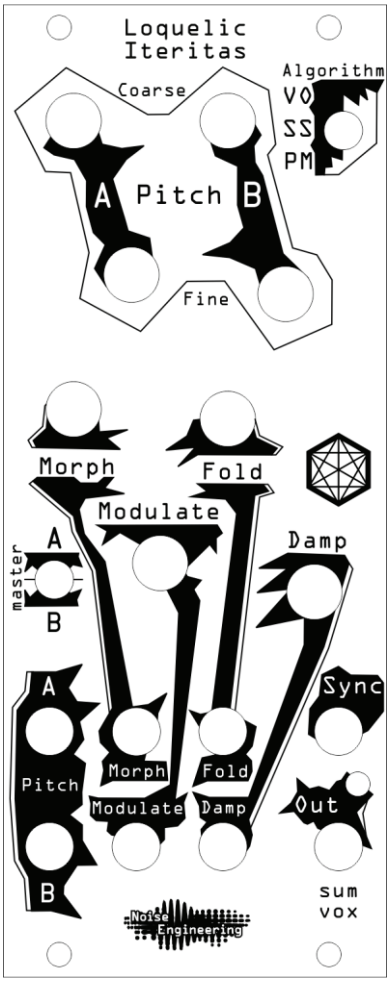
Noise Engineering

Noise Engineering

Loquelic Iteritas

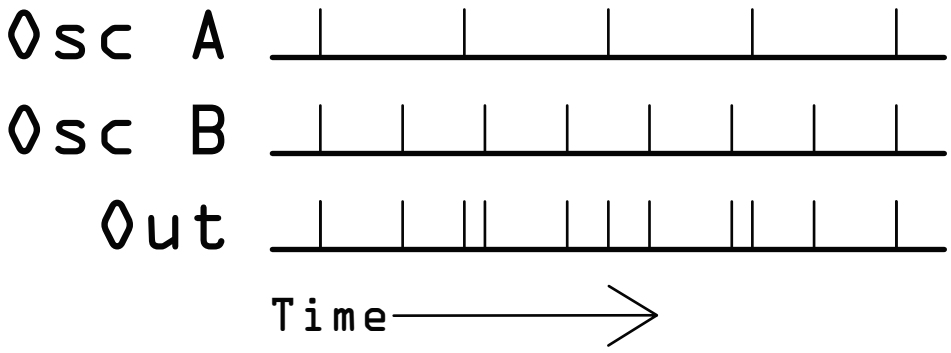
Complex Digital Oscilator

Sample Rate サンプル・レート



Loquelic Iteritasはユニークなマルチ・サンプリング方式を採用することでエリアジングをより音楽的に利用します。特定のサンプル・レートを選択することで倍音構造(全ての上音は基音の整数倍数です)を持つ波形とそのエリアジングを基音の倍数の周波数に動かすことで音楽的な鳴りを作り出します。

この方式は同一のDACを使用しているにも関わらず、2機のオシレーターを異なるピッチでシンセサイズする際に非常に複雑なものとなります。
Loquelic Iteritasにおけるその折衷案は固定サンプル・レートをやめて、
両オシレーターのサンプル基底する遅延時間を計算することです。
単一オシレーター機構の場合、この遅延はピッチにおいて完全に基底されます。
もしこの遅延が各オシレーターのそれぞれのピッチにおいて基底され、計算された場合、
両サンプル・レートがどちらのオシレーターの遅延が先に上がるかによって交互配置されます。
この時オシレーターは次のタイムステップへ更新され、出力値は両オシレーターの
出力段階によって基底、算出されます。
これはエリアジングがどこへ向かうかを正確に保障するものでは全くありません。
エリアジングを基音ピッチへ幾つかの方法で関連させるための試みであります。



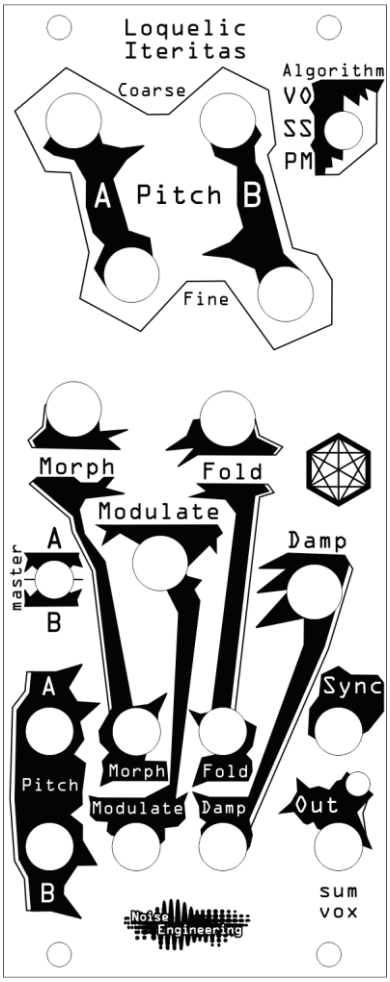
2つの独立したサンプル・レートを組み合わせることで
1つの不規則なサンプル・レートを形成します。
よってサンプル・レートは一定ではありません。

Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Design Notes デザイン・ノーツ



Loquelic Iteritasは2年の歳月をかけて開発されました。
Basimilus Iteritasと同時進行でデザインを始めましたがこちらの方が完成まで遥かに
時間がかかりました。当初のアイデアはVOSIMアルゴリズムの単純な実装化でしたが、
すぐに2つの付加的アルゴリズムの要素を割り込ませることが可能だと理解したのです。
Loquelic Iteritasはサウンド・デザイナー、ミュージシャンのための機能的なオシレーターとして
デザインされました。
私はシンプルで穏やかなサウンドから極端なものまで、さらには壊れたようなサウンドまで、
コントロール入力を配備したサウンド装置の可能性を最大限に引き出したいと考えました。
音色の多様性を重視したことによって全体のピッチ・レンジなどの音楽的側面は
犠牲にせざる終えませんでした。

使用されているアルゴリズムは極めてシンプルなもので、非常に興味深い未知の領域を
含んでいます。例えばPMモードでは激しい変調によって調整されることで
半分のサンプル・レートのえげつない自己発振サウンドを生成します。
不規則なサンプル・レートと組み合わせることで非常に興味深い、
しかしながら極めてうるさいサウンドを得ることができるでしょう。

Noise Engineering

Noise Engineering

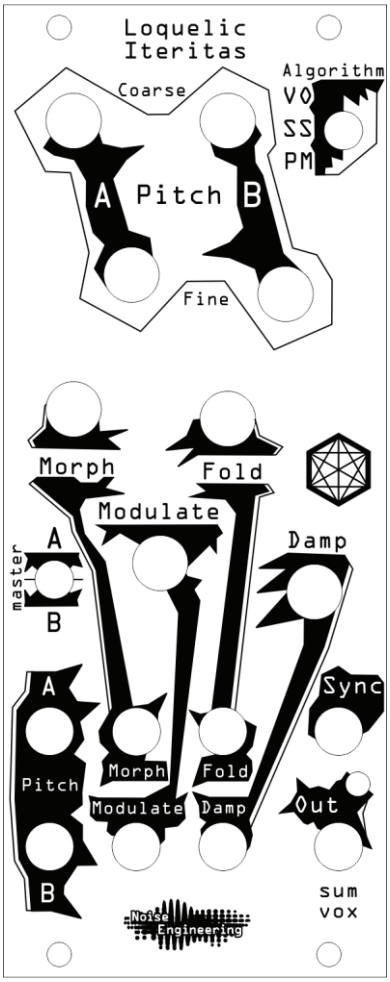
Loquelic Iteritas

Complex Digital Oscilator

Code コード

参照のために各アルゴリズムの中枢であるシンセジス・コードを記載しておきます。
これほどの多様なサウンドをこのような単純なアルゴリズムによって生成できることに
私は常に驚嘆しています。また他の皆さんもこのシンプルな美学に感動して頂ければ幸いです。

＊アルゴリズムの中枢以外の不要なコードは予め削除されています。



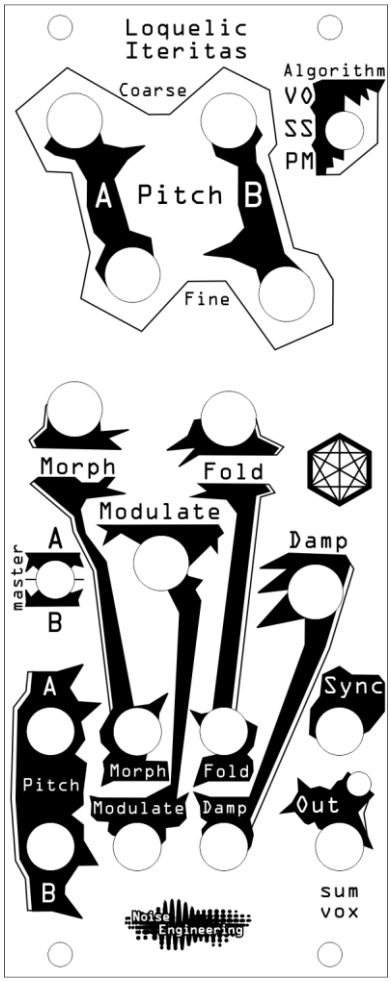
Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Code: V0

```
unsigned LI_FrameVO()
{
    int delay;

    if((state.voOsc.delay - state.voR1) < (state.voEnv.delay - state.voR2))
    {
        if(state.voOsc.sync && state.current.syncSw == LI_SYNC_B)
        {
            NeAttackDecayReset(state.voEnv);
        }

        state.voOutC = NeMoscSample(state.voOsc, state.morph, state.voMod);
        delay = state.voOsc.delay - state.voR1;
        if(delay < 0) delay = 0;
        state.voR1 = 0;
        state.voR2 += delay;
    }
    else
    {
        state.voOutE = NeAttackDecayOscSample(state.voEnv);
        state.voMod = fix24_mul(state.voModAmt, 2 * (state.voOutE - FIX24_HALF));

        if(state.voEnv.reset && state.current.syncSw == LI_SYNC_A)
        {
            NeMoscReset(state.voOsc);
        }

        delay = state.voEnv.delay -  state.voR2;
        if(delay < 0) delay = 0;
        state.voR2 = 0;
        state.voR1 += delay;
    }

    fix24 out = 0;
    out = NeFoldSample(state.fold, state.voOutC);
    out = fix24_mul(state.voOutE, out);
    out = fix24_mul(state.voMComp,out);
    out = fix24_soft_clip_poly(out);
    return fix24_to_u16_audio_delay(out, delay);
}
```

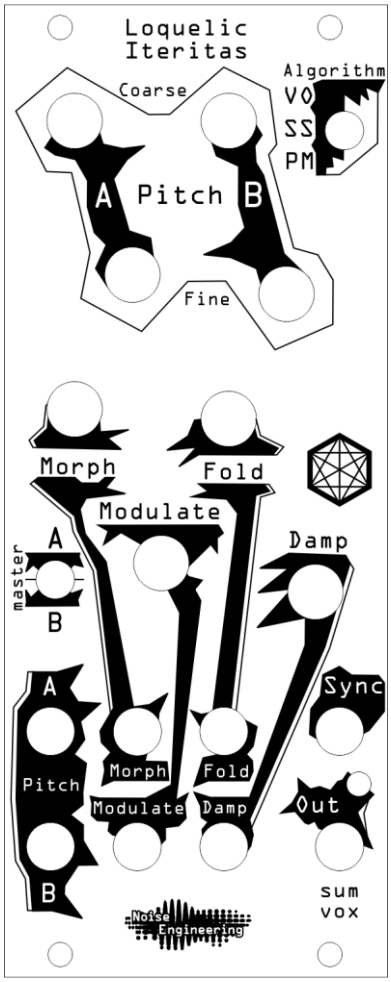


Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Code: DS



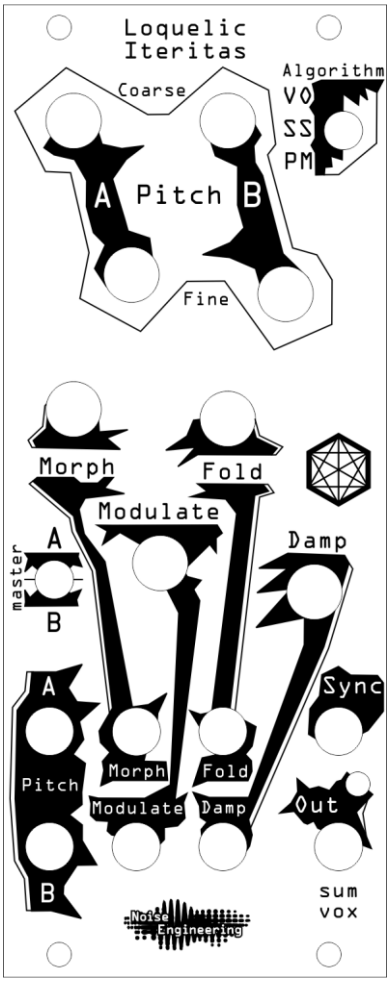

```
unsigned LI_FrameDS()
{
    fix24 out = 0;
    int delay = 0;

    state.dsPb = NextC( state.dsPc, state.dsPm, state.dsPb);

    int dc = state.dsOscC.delay - state.dsRc;
    int dm = state.dsOscM.delay - state.dsRm;
    int db = state.dsOscB.delay - state.dsRb;

    if(dc <= dm && dc <= db) //dc is next
    {
        fix24 phaseC = fix24_mul(state.dsOutM, state.dsMod);
        state.dsOutC = NeMoscSample(state.dsOscC, state.morph, phaseC);
        delay = dc;
        if(delay < 0) delay = 0;
        state.dsRc = -delay;
    }
    if(dm <= dc && dm <= db) //dm is next
    {
        fix24 phaseM = FIX24_QUARTER + state.morph;
        state.dsOutM = NeMoscSample(state.dsOscM, state.morph, phaseM);
        delay = dm;
        if(delay < 0) delay = 0;
        state.dsRm = -delay;
    }
    if(db <= dm && db <= dc) //db is next
    {
        state.dsOutB = NeMoscSample(state.dsOscB, state.morph);
        delay = db;
        if(delay < 0) delay = 0;
        state.dsRb = -delay;
    }

    if(state.current.syncSw == LI_SYNC_A)
    {
        if(state.dsOscM.sync) NeMoscReset(state.dsOscC);
    }
    else if(state.current.syncSw == LI_SYNC_B)
    {
        if(state.dsOscC.sync) NeMoscReset(state.dsOscM);
    }

    state.dsRc += delay;
    state.dsRm += delay;
    state.dsRb += delay;

    fix24 a = state.dsA;
    fix24 a2 = fix24_mul(a, a);
    fix24 n = state.dsOutC - fix24_mul(a, state.dsOutB);
    fix24 d = FIX24_128TH + FIX24_ONE + a2 - 2 * fix24_mul(a, state.dsOutM);
    out = fix24_mul(FIX24_3RD,fix24_div(n, d));
    out = fix24_mul(state.morphScale, out);
    out = fix24_soft_clip_poly(out);
    out = NeFoldSample(state.fold, out);
    return fix24_to_u16_audio_delay(out, 2 * delay);
}
```

Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Code: PM




```
unsigned LI_FramePM()
{
    fix24 out = 0;
    int delay = 0;

    int updateDelay1 = state.pmOsc1.delay - state.pmR1;
    int updateDelay2 = state.pmOsc2.delay - state.pmR2;

    if(updateDelay1 <= updateDelay2) //update whichever osc is due next
    {
        state.pmOut1 = NeMoscSample(state.pmOsc1, state.morph, state.pmPhase1);
        delay = updateDelay1;
        if(delay < 0) { delay = 0; }
        state.pmR1 = 0;
        state.pmR2 += delay;
    }
    else
    {
        state.pmOut2 = NeMoscSample(state.pmOsc2, state.morph, state.pmPhase2);
        delay = updateDelay2;
        if(delay < 0) { delay = 0; }
        state.pmR1 += delay;
        state.pmR2 = 0;
    }

    if(state.current.syncSw == LI_SYNC_A && state.pmOsc2.sync)
    {
        NeMoscReset(state.pmOsc1);
    }
    else if(state.current.syncSw == LI_SYNC_B && state.pmOsc1.sync)
    {
        NeMoscReset(state.pmOsc2);
    }

    state.pmPhase1 = (7 * state.pmPhase1 + fix24_mul(state.pmMod1, state.pmOut2))>>3;
    state.pmPhase2 = (7 * state.pmPhase2 + fix24_mul(state.pmMod2, state.pmOut1))>>3;

    fix24 am1 = fix24_mul(state.pmOut1, state.pmAM1);
    fix24 am2 = fix24_mul(state.pmOut2, state.pmAM2);
    fix24 am3 = fix24_mul(am1,am2);
    out = am1 + am2 + am3;
    out = fix24_soft_clip_poly(out);
    out = NeFoldSample(state.fold, out);
    return fix24_to_u16_audio_delay(out, delay);
}
```

Noise Engineering

Loquelic Iteritas

Complex Digital Oscilator

Special Thanks

Kris Kaiser
Shawn Jimmerson
Cyrus Makarechian
William Mathewson
Mickey Bakas
Tyler Thompson
Alex Anderson