```
#Define @NoteSector1Harmonics 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
#Define @NoteSector2Harmonics 20381,30596,31865,36363,37109,65535,32518,26007,23495,14430,12391,6875,6875,5432,5262,4753,4243,2376,1867,1867,0,0,0,0,0,0,0,0,0,0,0,0
#Define @NoteSector3Harmonics 33524,50325,65535,38647,16804,12019,7457,5262,4073,3734,3055,2716,1528,1528,678,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
#Define @NoteSector4Harmonics 65535,50841,18227,7766,5262,3564,3225,2716,2376,1697,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
#Define @CentreWaveformHarmonicScaling 1.000,1.500,1.500,1.500,1.500,1.100,1.100,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000
#Define @UpperIntensityLayerHarmonicScaling 1.000,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500,1.500

//*******************************************************************************
//      TRUMPET VERSION 1.0 Jeremy Leach 25th October 2023
//*******************************************************************************

// IMPORTANT:
//==========

// Remember to enable the 'Harmonic pruning' option in Tools>Settings to benefit from faster timbre refresh rate.

// You MUST have version 10 (or above) of the Tone Processor firmware loaded to use :
// 1. The Vibrato with 'Fluctuating' option. If you don't have this version loaded then disable Vibrato (unless you think normal Vibrato enhances the sound).
// 2. The 'Fluctuating' WaveType in the Vibrato and Timbre Morth LFO.

// You must have version 11 (or above) of the Tone Processor firmware loaded to :
// 1. Enjoy EQed noise that sounds far closer to blown noise than just white noise.
// 2. Enjoy noise being added to a tone    BEFORE tremolo modulation, so that the noise level is modulated with the amplitude. This    gives more realistic blown sounds.

// You must have version 12 (or above) of the Tone Processor firmware loaded to :
// 1. Benefit from Improved amplitude envelope interpolation.

// You must have version 14 (or above) of the Tone Processor firmware loaded to :
// 1. Benefit from legato.
// 2. Benefit from corrected 'fluctuation' range.

//EXAMPLE NOTE SEQUENCES FOR TESTING PURPOSES
//==================================================================================
//Blue Skies - Irvin Berlin.      Example : https://www.youtube.com/shorts/3Be7Ab54NBs
//NOTE: The note sequencer isn't really designed to play notes with any complex timing as in this song, so it's a best approximation ! Play at tempo of 960.
//[+E4][-E4][R7][+B4][R2]
//[R23][-B4]
//[+A4][R2][~G4,-A4][][~A4,-G4][R6]
//[~B4,-A4][R32][-B4][R2]
//[+A4][R2][~G4,-A4][][~A4,-G4][R6]
//[+B4,-A4][-B4][R7]
//[+D4][R23][-D4][R2]
//[+D4][R2][~E4,-D4][R2][-E4][R5]
//[+G4][R15][~A4,-G4][R2][~G4,-A4][R4]
//[~F#4,-G4][R22][-F#4]


// INFORMATION ABOUT A TRUMPET TONE :
//=============================
//Overall : The acoustics of Trumpets is hugely complex, and very hard to model comprehensively.    Detailed website here : https://newt.phys.unsw.edu.au/jw/brassacoustics.html
//There are different types of Trumpet, but broadly speaking the commonest Bb Trumpet has a note range of MIDI note F#3 to C6, but can go higher. This corresponds to note sector 2 (F3 to G#4) to    4 (C#6 and above) on this module.
//Players DON'T overblow to raise an octave, but by changing the position of his lips on the mouthpiece or pushing down one or more or the valves (the part of the tubing that is sounding is made longer so the pitch changes).
//The brass instrument is a 'closed' pipe, The instrument is open at the far end or bell. But it is (almost) closed at the other end.
//A player's lips vibrate at one particular frequency. The harmonics in the lip vibration set up, and are usually in turn reinforced by, standing waves, because the instrument is designed to produce standing waves with harmonic frequency ratios.
//For brass instruments, the radiated power sometimes increases with frequency over the low part of the range, as here. This means that one tends to get maximum radiation at a moderately high frequency (typically around several hundred Hz,
//although it has a different value in each lip reed instrument), and less power at lower and higher frequencies. Further, the enclosed air in the mouthpiece and the constriction beyond it also tend to drive the lips most efficiently near this frequency,
//Open-closed pipes produce sound with odd harmonics. However it's more complicated than this and multiple factors come into play.
//Harmonics are 'stretched', as a result of the complex physics involved, resulting in inharmonicity which is similar to that in strings, but due to complex air-related reasons rather than string stiffness.
//The harmonic spectrum depends strongly on how loudly you play. The larger the vibration of the lips, the more non-linear the vibration, so the more strong high harmonics present in the sound

// GENERAL APPROACH:
//===============
//1. After trying to use the instrument Analyser, it didn't work well so instead analysed real trumpet samples, in Audacity, for notes corresponding to the note sectors. Both looking at spectrum and amplitude envelope.
//2. Paid particular attention to the initial 'Parp' sound, the low blown noise, and the sharp release.
//3. Used a new 'Legato' patch option to allow sequential notes to sound more realistic, by only having the attack 'parp' in the first note.
//4. Mellowed the low intensities, brightened the high intensities and added in some randomness across the timbral lanscape that the Timbre LFO can fluctuate across.
//5. Used 'fluctuation' of timbre, amplitude and frequency, in the 3 LFOs for timbre, tremolo and vibrato. This seems pretty effective to emulate the wobbliness of playing this very dynamic and sonically turbulent instrument.
//6. Added an almost imperceptable amount of sample for the attack and release phases, because it seems to add some realism.
//7. Applied a little bandpass boost in the Filter between 256Hz to 323Hz because this is where the maximum energy transfer occurs from the player's lips.
//8. I couldn't see a reason for any key-scaling, so didn't use any !


//Optional first step to set the patch to a default state.
//Run("\ClearAll.txt");

Define.WaveSet
{
        //We use example harmonic levels from the reasearch paper pdf mentioned above, for notes F3,F4,F5,F6

        //First clear all harmonic levels in the current WaveSet.
        WaveSet.SetCurrentNoteSector(NoteSector:0);
```

```
    WaveSet.SetCurrentIntensityLayer(IntensityLayer:0);
    CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
    CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:@NoteSector1Harmonics);
    CurrentWaveform.CopyAcrossAllDimensions();

    //Select note sector 1    (C#2 to E3) DOES NOT APPLY TO TRUMPET
    WaveSet.SetCurrentNoteSector(NoteSector:1);
    WaveSet.SetCurrentIntensityLayer(IntensityLayer:0);
    CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
    CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:@NoteSector1Harmonics);
    //Copy this waveform to all others in the current waveform block
    CurrentWaveform.CopyToRange(WaveformFrom:0,WaveformTo:4);
    CurrentWaveform.ScaleHarmonicLevelsFromCSV(ScaleCSV:@CentreWaveformHarmonicScaling);

    //Select note sector 2    (F3 to G#4). Harmonics taken from real Bb trumpet sample playing G3.
    WaveSet.SetCurrentNoteSector(NoteSector:2);
    //Set the harmonic levels to levels analysed in Audacity Spectrum analyser, for a real sample of a Trumpet playing note F3.
    CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:@NoteSector2Harmonics);
    //Copy this waveform to all others in the current waveform block
    CurrentWaveform.CopyToRange(WaveformFrom:0,WaveformTo:4);
    CurrentWaveform.ScaleHarmonicLevelsFromCSV(ScaleCSV:@CentreWaveformHarmonicScaling);

    //Select note sector 3 (A4 to C6). Harmonics taken from real Bb trumpet sample playing A4.
    WaveSet.SetCurrentNoteSector(NoteSector:3);
    //Set the harmonic levels to levels analysed in Audacity Spectrum analyser, for a real sample of a Trumpet playing note A4.
    CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:@NoteSector3Harmonics);
    //Copy this waveform to all others in the current waveform block
    CurrentWaveform.CopyToRange(WaveformFrom:0,WaveformTo:4);
    CurrentWaveform.ScaleHarmonicLevelsFromCSV(ScaleCSV:@CentreWaveformHarmonicScaling);

    //Select    note sector 4 (C#6 and above). Harmonics taken from real Bb trumpet sample playing G5.
    WaveSet.SetCurrentNoteSector(NoteSector:4);
    //Set the harmonic levels to levels analysed in Audacity Spectrum analyser, for a real sample of a Trumpet playing note C6 (Sample was actually C6 and not C#6, but it makes no real difference!).
    CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:@NoteSector4Harmonics);
    //Copy this waveform to all others in the current waveform block
    CurrentWaveform.CopyToRange(WaveformFrom:0,WaveformTo:4);
    CurrentWaveform.ScaleHarmonicLevelsFromCSV(ScaleCSV:@CentreWaveformHarmonicScaling);


    //All of the above has been on intensity layer 0, so now initially copy everything to the intensity layers above
    CurrentIntensityLayer.CopyToRange(IntensityLayerFrom:0,IntensityLayerTo:2);

    //For intensity layer 0 , we want to mute the high frequencies quite a lot, because when the Trumpet is played softly there is less energy being delivered to the instrument , and since high frequency energy gets damped faster then this is effectively low-pass filtering.
    CurrentIntensityLayer.ShapeTheHarmonics(Direction:Up,HarmonicIDFrom:1,Slope:-6dB);

    //For the highest intensity layer, boost the upper harmonics. This has the effect of raising it an octave and making it quite shrill.
    WaveSet.SetCurrentIntensityLayer(IntensityLayer:2);
    CurrentIntensityLayer.ScaleHarmonicLevelsFromCSV(ScaleCSV:@UpperIntensityLayerHarmonicScaling);

    //Significantly reduce the overall gain of the highest note sector (C#6 and above), to make it less shrill
    WaveSet.HighestNoteSectorGain(50%);

    //We sprinkle in some randomness into the Harmonic levels across the timbral landscape, to add authenticity.
    //This also means that all waveforms are noticeably different when we apply slow timbral modulation between them.
    WaveSet.AddHarmonicRandomness(25%);
}

Define.Filter
{
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:0,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:1,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:2,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:3,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:4,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:5,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:6,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:7,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:8,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:9,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:10,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:11,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:12,Level:4);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:13,Level:4);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:14,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:15,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:16,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:17,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:18,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:19,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:20,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:21,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:22,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:23,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:24,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:25,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:26,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:27,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:28,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:29,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:30,Level:0);
    CurrentFilter.SetFilterBandLevel(FilterBandNumber:31,Level:0);
}

Define.Patch
{
    //===== GENERAL : Misc ==========
    General.SetDetuning(DetuningType:Random,Detuning:0%);
    General.SetDetuning(DetuningType:Regular,Detuning:0%);
    General.SetHarmonicAlgorithm(HarmonicAlgorithm:RailsbackInharmonicity);
    General.SetPortamento(Enabled:False,PortamentoAmount:100%);
    General.SetLegato(Enabled:True);
    General.SetScalingSplit(Note:f#4);
    General.SetVelocityCurve(Curve:Normal);

    //===== GENERAL : Oscillators ==========
    General.SetActiveOscillators(OscillatorCount:1);
    General.SetDetuningMode(DetuningMode:Hz);
    General.SetOscillatorDetuning(Oscillator:0,Detuning:0.00);
    General.SetOscillatorDetuning(Oscillator:1,Detuning:0.40);

    //===== GENERAL : Envelope control ==========
    General.SetEnvelopeGainController(Envelope:Amplitude,InitialLevel:0.0,GainCC:Velocity);
    General.SetEnvelopeGainController(Envelope:NoiseGain,InitialLevel:20.0,GainCC:None);
    General.SetEnvelopeGainController(Envelope:NoiseCutoffFrequency,InitialLevel:100.0,GainCC:None);
    General.SetEnvelopeGainController(Envelope:TimbreMorph,InitialLevel:0.0,GainCC:None);
    General.SetEnvelopeGainController(Envelope:SampleGain,InitialLevel:1.5,GainCC:Velocity);
    General.SetEnvelopeGainController(Envelope:PitchShift,InitialLevel:0.0,GainCC:None);
    General.SetEnvelopeGainController(Envelope:TremoloDepth,InitialLevel:60.0,GainCC:None);
    General.SetEnvelopeGainController(Envelope:VibratoDepth,InitialLevel:0.0,GainCC:None);
    General.SetEnvelopeGainController(Envelope:TimbreLFODepth,InitialLevel:0.0,GainCC:None);

    //===== GENERAL : LFOs ==========
    General.SetLFO(LFOType:Tremolo,Enabled:True,WaveType:Fluctuating,Frequency:3.00,FrequencyCC:None,DepthCC:None);
    General.SetLFO(LFOType:Vibrato,Enabled:True,WaveType:Fluctuating,Frequency:0.00,FrequencyCC:None,DepthCC:None);
    General.SetLFO(LFOType:TimbreLFO,Enabled:True,WaveType:Fluctuating,Frequency:3.00,FrequencyCC:None,DepthCC:None);

    //===== ADSR SECTIONS ==========
    ADSR.ConfigureSection(Section:Attack,Enabled:True,Duration:25,EndKSU:0.000%,EndKSL:0.000%,Sample:KeyClick,SampleMode:OneShot);
    ADSR.ConfigureSection(Section:Decay,Enabled:True,Duration:40,EndKSU:0.000%,EndKSL:0.000%,Sample:None,SampleMode:OneShot);
    ADSR.ConfigureSection(Section:Sustain,Enabled:True,Duration:20000,EndKSU:0.000%,EndKSL:0.000%,Sample:None,SampleMode:OneShot);
    ADSR.ConfigureSection(Section:Release,Enabled:True,Duration:150,EndKSU:0.000%,EndKSL:0.000%,Sample:KeyClick,SampleMode:OneShot);

    //----- ENVELOPES : for ADSR section 'Attack' ----------
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:Amplitude,EnvelopeType:Linear,Target:85.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:1750,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:20.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:NoiseGain,EnvelopeType:Linear,Target:1.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:1000,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:NoiseCutoffFrequency,EnvelopeType:None,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:TimbreMorph,EnvelopeType:Linear,Target:25.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:15000,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:SampleGain,EnvelopeType:Linear,Target:1.50%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:500,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:PitchShift,EnvelopeType:None,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:TremoloDepth,EnvelopeType:Linear,Target:55.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:1000,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:VibratoDepth,EnvelopeType:Linear,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:TimbreLFODepth,EnvelopeType:Linear,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);

    //----- ENVELOPES : for ADSR section 'Decay' ----------
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:Amplitude,EnvelopeType:Linear,Target:33.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:900,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:20.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:NoiseGain,EnvelopeType:Linear,Target:1.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:1000,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:NoiseCutoffFrequency,EnvelopeType:None,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:TimbreMorph,EnvelopeType:Linear,Target:49.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:15000,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:SampleGain,EnvelopeType:Linear,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:500,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:PitchShift,EnvelopeType:None,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:TremoloDepth,EnvelopeType:Linear,Target:55.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:1000,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:VibratoDepth,EnvelopeType:Linear,Target:30.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:5000,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:TimbreLFODepth,EnvelopeType:Linear,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:10000,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);

    //----- ENVELOPES : for ADSR section 'Sustain' ----------
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:Amplitude,EnvelopeType:Linear,Target:100.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:1000,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:0.038%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:NoiseGain,EnvelopeType:Linear,Target:1.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:NoiseCutoffFrequency,EnvelopeType:None,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:TimbreMorph,EnvelopeType:Linear,Target:50.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:1,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:SampleGain,EnvelopeType:Linear,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:PitchShift,EnvelopeType:None,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:TremoloDepth,EnvelopeType:Linear,Target:55.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:1000,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:VibratoDepth,EnvelopeType:Linear,Target:30.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:5000,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:TimbreLFODepth,EnvelopeType:Linear,Target:100.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:4000,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);

    //----- ENVELOPES : for ADSR section 'Release' ----------
    ADSR.ConfigureEnvelope(Section:Release,Envelope:Amplitude,EnvelopeType:Linear,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:350,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:0.153%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:NoiseGain,EnvelopeType:Linear,Target:1.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:NoiseCutoffFrequency,EnvelopeType:None,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:TimbreMorph,EnvelopeType:Linear,Target:75.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:1500,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:SampleGain,EnvelopeType:Linear,Target:0.25%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:500,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:PitchShift,EnvelopeType:None,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:TremoloDepth,EnvelopeType:Linear,Target:55.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:VibratoDepth,EnvelopeType:Linear,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:TimbreLFODepth,EnvelopeType:Linear,Target:0.00%,TargetKSU:0.000%,TargetKSL:0.000%,LinearDelta:0,LinearDeltaKSU:0.000%,LinearDeltaKSL:0.000%,ExpMult:3.000%,ExpMultKSU:0.000%,ExpMultKSL:0.000%);
}
```