```
//*****************************************************************************
//      FLUTE 2.3 Jeremy Leach 20th September 2023
//*****************************************************************************

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

//Use modulation wheel to vary tremolo and Timbre LFO depth.
//Velocity matters here, and a velocity of around 110 gives a lot of 'breath' noise.

//This is a useful test note sequence in the note Sequencer (The Ash Grove, a traditional Welsh folk song). Set tempo to 220. :
//[+g4][-g4][+c5][-c5][+e5][-e5][g5][f5][+e5][-e5][+c5][-c5][+c5][-c5][+d5][-d5][f5][e5][d5][c5][+b4][-b4][+g4][-g4][+g4][-g4][+c5][-c5][e5][d5][c5][b4][+a4][-a4][+f4][-f4][+a4][-a4][+g4][-g4][+c5][-c5][+b4][-b4][+c5][][][][-c5]

//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 'Boundary' velocity curve.
//3. The 'Fluctuating' WaveType in the Vibrato.

//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.

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

// INFORMATION ABOUT A FLUTE TONE :
//============================
//Overall : The acoustics of flutes is hugely complex, and very hard to model comprehensively.  This website gives a flavour of how complex : https://newt.phys.unsw.edu.au/jw/fluteacoustics.html
//A flute has a note range of MIDI note C4  to C7. This corresponds to note sector 2 (F3 to G#4) to  4 (C#6 and above) on this module.
//Flutes, are effectively open pipes, because they have the blow hole and the exit hole.
//The player blows into the mouthpiece and an edge creates turbulent pulses of air that set up a standing wave within the pipe.
//Open pipes produce sound with all the harmonics in the harmonic series (unlike closed pipes like the Clarinet, that only include the odd harmonics).
//If you over-blow a flute it jumps up a mode, so that the second harmonic becomes the lowest pitch produced.
//The physical flute has an overall frequency response with a cut-off around 2KHz.
//Inaccuracies of playing coupled with the complexities of open and closed finguring combinations make notes often a little sharp or flat.
//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 harder you blow, generally the more spectral content you introduce in the upper harmonics.
//There is an element of noise due to blowing, but it's debateable whether this should be modelled because a listener from a distance might find this element insignicant. Also the noise has resonant peaks and isn't just white noise !


// GENERAL APPROACH:
//================
//In this patch, we're going to :
//1. USE THE INSTRUMENT ANALAYSER to analyse real flute samples, in terms of the WaveSet and envelope, because this gives much better results than trying to make educated guesses.
//2. BUT we also want to emulate overblowing at max intensity, so at max intensity set the fundamental harmonic level to zero (interesting!).
//3. Set the filter as low-pass with  a cut-off around 2KHz and fairly gentle slope.
//4. Set the harmonic algorithm to 'Railsback Inharmonicity', even though we're not emulating strings.
//5. Add a tiny amount of delayed Tremolo and Timbre morphing LFO, to emulate the slightly unsteady air flow.
//6. Add a small amount of noise to emulate the blown air.
//7. Add a fair amount of pitch randomness to emulate the wobbly nature of  playing and pitch inaccuracy through fingering.


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

Define.WaveSet
{
        //This long block of WaveSet definition is from using the Instrument Analyser to analyze a good set of sampled Flute waveforms available online at https://freesound.org/people/MTG/packs/20209/
        //=============================================================================================================================================

        WaveSet.SetCurrentNoteSector(NoteSector:0);
        WaveSet.SetCurrentIntensityLayer(IntensityLayer:0);
        CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);

        //##### NOTE SECTOR 0 ##############################################
        WaveSet.SetCurrentNoteSector(NoteSector:0);
```

```
//===== INTENSITY LAYER    0 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:0);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,58185,14434,8562,20786,4432,14178,2108,2898,2964,1389,649,811,521,254,154,206,206,79,103,129,99,60,72,82,71,59,58,37,45,32,35);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:59895,65535,21256,13698,16836,6194,20445,2100,1813,3151,1830,609,559,488,285,130,211,267,104,74,104,150,86,67,63,74,62,52,56,70,41,35);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//===== INTENSITY LAYER    1 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:1);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,58185,14434,8562,20786,4432,14178,2108,2898,2964,1389,649,811,521,254,154,206,206,79,103,129,99,60,72,82,71,59,58,37,45,32,35);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:59895,65535,21256,13698,16836,6194,20445,2100,1813,3151,1830,609,559,488,285,130,211,267,104,74,104,150,86,67,63,74,62,52,56,70,41,35);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//===== INTENSITY LAYER    2 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:2);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,58185,14434,8562,20786,4432,14178,2108,2898,2964,1389,649,811,521,254,154,206,206,79,103,129,99,60,72,82,71,59,58,37,45,32,35);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:59895,65535,21256,13698,16836,6194,20445,2100,1813,3151,1830,609,559,488,285,130,211,267,104,74,104,150,86,67,63,74,62,52,56,70,41,35);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
```

```
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//##### NOTE SECTOR 1 ############################################
WaveSet.SetCurrentNoteSector(NoteSector:1);

//===== INTENSITY LAYER    0 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:0);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,58185,14434,8562,20786,4432,14178,2108,2898,2964,1389,649,811,521,254,154,206,206,79,103,129,99,60,72,82,71,59,58,37,45,32,35);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:59895,65535,21256,13698,16836,6194,20445,2100,1813,3151,1830,609,559,488,285,130,211,267,104,74,104,150,86,67,63,74,62,52,56,70,41,35);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//===== INTENSITY LAYER    1 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:1);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,58185,14434,8562,20786,4432,14178,2108,2898,2964,1389,649,811,521,254,154,206,206,79,103,129,99,60,72,82,71,59,58,37,45,32,35);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:59895,65535,21256,13698,16836,6194,20445,2100,1813,3151,1830,609,559,488,285,130,211,267,104,74,104,150,86,67,63,74,62,52,56,70,41,35);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//===== INTENSITY LAYER    2 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:2);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,58185,14434,8562,20786,4432,14178,2108,2898,2964,1389,649,811,521,254,154,206,206,79,103,129,99,60,72,82,71,59,58,37,45,32,35);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:59895,65535,21256,13698,16836,6194,20445,2100,1813,3151,1830,609,559,488,285,130,211,267,104,74,104,150,86,67,63,74,62,52,56,70,41,35);

//----- Waveform    3 ----------
```

```
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//##### NOTE SECTOR 2 ################################################
WaveSet.SetCurrentNoteSector(NoteSector:2);

//===== INTENSITY LAYER    0 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:0);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,58185,14434,8562,20786,4432,14178,2108,2898,2964,1389,649,811,521,254,154,206,206,79,103,129,99,60,72,82,71,59,58,37,45,32,35);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:59895,65535,21256,13698,16836,6194,20445,2100,1813,3151,1830,609,559,488,285,130,211,267,104,74,104,150,86,67,63,74,62,52,56,70,41,35);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//===== INTENSITY LAYER    1 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:1);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,58185,14434,8562,20786,4432,14178,2108,2898,2964,1389,649,811,521,254,154,206,206,79,103,129,99,60,72,82,71,59,58,37,45,32,35);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:59895,65535,21256,13698,16836,6194,20445,2100,1813,3151,1830,609,559,488,285,130,211,267,104,74,104,150,86,67,63,74,62,52,56,70,41,35);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//===== INTENSITY LAYER    2 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:2);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,58185,14434,8562,20786,4432,14178,2108,2898,2964,1389,649,811,521,254,154,206,206,79,103,129,99,60,72,82,71,59,58,37,45,32,35);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);
```

```
//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:59895,65535,21256,13698,16836,6194,20445,2100,1813,3151,1830,609,559,488,285,130,211,267,104,74,104,150,86,67,63,74,62,52,56,70,41,35);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:58283,65535,17981,17589,11556,5562,16514,2940,446,1611,1277,792,201,167,118,160,140,233,92,98,141,298,56,132,40,33,26,70,24,71,43,24);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//##### NOTE SECTOR 3 ############################################
WaveSet.SetCurrentNoteSector(NoteSector:3);

//===== INTENSITY LAYER    0 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:0);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,10915,10307,4768,4128,3588,596,705,586,284,207,171,110,291,94,84,81,72,55,53,52,47,32,40,32,28,20,32,33,22,14,16);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,11802,7227,2927,1029,681,355,220,193,166,124,94,115,78,64,78,72,61,56,53,60,50,44,45,41,35,40,37,40,36,24,22);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,16057,12447,4526,3051,2066,550,537,386,198,177,162,166,281,201,108,74,79,69,59,56,54,40,49,51,39,49,42,35,26,19,21);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,11761,6368,3919,1186,745,341,256,197,118,83,97,162,237,165,68,61,56,49,52,39,37,32,43,41,25,34,37,34,20,15,17);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//===== INTENSITY LAYER    1 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:1);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,10915,10307,4768,4128,3588,596,705,586,284,207,171,110,291,94,84,81,72,55,53,52,47,32,40,32,28,20,32,33,22,14,16);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,11802,7227,2927,1029,681,355,220,193,166,124,94,115,78,64,78,72,61,56,53,60,50,44,45,41,35,40,37,40,36,24,22);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,16057,12447,4526,3051,2066,550,537,386,198,177,162,166,281,201,108,74,79,69,59,56,54,40,49,51,39,49,42,35,26,19,21);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,11761,6368,3919,1186,745,341,256,197,118,83,97,162,237,165,68,61,56,49,52,39,37,32,43,41,25,34,37,34,20,15,17);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//===== INTENSITY LAYER    2 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:2);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,10915,10307,4768,4128,3588,596,705,586,284,207,171,110,291,94,84,81,72,55,53,52,47,32,40,32,28,20,32,33,22,14,16);
```

```
//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,11802,7227,2927,1029,681,355,220,193,166,124,94,115,78,64,78,72,61,56,53,60,50,44,45,41,35,40,37,40,36,24,22);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,16057,12447,4526,3051,2066,550,537,386,198,177,162,166,281,201,108,74,79,69,59,56,54,40,49,51,39,49,42,35,26,19,21);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,11761,6368,3919,1186,745,341,256,197,118,83,97,162,237,165,68,61,56,49,52,39,37,32,43,41,25,34,37,34,20,15,17);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//##### NOTE SECTOR 4 ################################################
WaveSet.SetCurrentNoteSector(NoteSector:4);

//===== INTENSITY LAYER    0 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:0);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,7159,1911,754,423,813,1044,315,385,167,253,193,364,127,72,43,38,33,44,0,0,0,0,0,0,0,0,0,0,0,0);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,7290,1957,731,400,533,1660,226,147,186,175,163,420,112,70,46,28,32,41,0,0,0,0,0,0,0,0,0,0,0,0,0);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,6291,2286,652,447,801,1178,355,290,166,193,170,385,112,96,34,28,41,38,0,0,0,0,0,0,0,0,0,0,0,0,0);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,5915,1983,723,430,581,1394,335,198,208,129,96,97,137,49,28,34,14,13,0,0,0,0,0,0,0,0,0,0,0,0,0);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//===== INTENSITY LAYER    1 ==========
WaveSet.SetCurrentIntensityLayer(IntensityLayer:1);

//----- Waveform    0 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,7159,1911,754,423,813,1044,315,385,167,253,193,364,127,72,43,38,33,44,0,0,0,0,0,0,0,0,0,0,0,0,0);

//----- Waveform    1 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,7290,1957,731,400,533,1660,226,147,186,175,163,420,112,70,46,28,32,41,0,0,0,0,0,0,0,0,0,0,0,0,0);

//----- Waveform    2 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,6291,2286,652,447,801,1178,355,290,166,193,170,385,112,96,34,28,41,38,0,0,0,0,0,0,0,0,0,0,0,0,0);

//----- Waveform    3 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,5915,1983,723,430,581,1394,335,198,208,129,96,97,137,49,28,34,14,13,0,0,0,0,0,0,0,0,0,0,0,0,0);

//----- Waveform    4 ----------
CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV: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);

//===== INTENSITY LAYER    2 ==========
```

```
        WaveSet.SetCurrentIntensityLayer(IntensityLayer:2);

        //----- Waveform    0 ----------
        CurrentWaveformBlock.SetCurrentWaveform(Waveform:0);
        CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,7159,1911,754,423,813,1044,315,385,167,253,193,364,127,72,43,38,33,44,0,0,0,0,0,0,0,0,0,0,0,0);

        //----- Waveform    1 ----------
        CurrentWaveformBlock.SetCurrentWaveform(Waveform:1);
        CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,7290,1957,731,400,533,1660,226,147,186,175,163,420,112,70,46,28,32,41,0,0,0,0,0,0,0,0,0,0,0,0);

        //----- Waveform    2 ----------
        CurrentWaveformBlock.SetCurrentWaveform(Waveform:2);
        CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,6291,2286,652,447,801,1178,355,290,166,193,170,385,112,96,34,28,41,38,0,0,0,0,0,0,0,0,0,0,0,0);

        //----- Waveform    3 ----------
        CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);
        CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:65535,5915,1983,723,430,581,1394,335,198,208,129,96,97,137,49,28,34,14,13,0,0,0,0,0,0,0,0,0,0,0,0);

        //----- Waveform    4 ----------
        CurrentWaveformBlock.SetCurrentWaveform(Waveform:4);
        CurrentWaveform.SetHarmonicLevelsFromCSV(LevelCSV:0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
}

Define.WaveSet
{
        //We now perform additional manipulation to the WaveSet that has been derived from the Intrument Analyser samples.

        //The Instrument Analyser doesn't populate Waveform4, so lets copy Waveform3 to Waveform4 for the relevant waveform blocks:
        CurrentWaveformBlock.SetCurrentWaveform(Waveform:3);

        WaveSet.SetCurrentNoteSector(NoteSector:2);
        WaveSet.SetCurrentIntensityLayer(IntensityLayer:0);
        CurrentWaveform.CopyTo(WaveformTo:4);
        WaveSet.SetCurrentIntensityLayer(IntensityLayer:1);
        CurrentWaveform.CopyTo(WaveformTo:4);
        WaveSet.SetCurrentIntensityLayer(IntensityLayer:2);
        CurrentWaveform.CopyTo(WaveformTo:4);

        WaveSet.SetCurrentNoteSector(NoteSector:3);
        WaveSet.SetCurrentIntensityLayer(IntensityLayer:0);
        CurrentWaveform.CopyTo(WaveformTo:4);
        WaveSet.SetCurrentIntensityLayer(IntensityLayer:1);
        CurrentWaveform.CopyTo(WaveformTo:4);
        WaveSet.SetCurrentIntensityLayer(IntensityLayer:2);
        CurrentWaveform.CopyTo(WaveformTo:4);

        WaveSet.SetCurrentNoteSector(NoteSector:4);
        WaveSet.SetCurrentIntensityLayer(IntensityLayer:0);
        CurrentWaveform.CopyTo(WaveformTo:4);
        WaveSet.SetCurrentIntensityLayer(IntensityLayer:1);
        CurrentWaveform.CopyTo(WaveformTo:4);
        WaveSet.SetCurrentIntensityLayer(IntensityLayer:2);
        CurrentWaveform.CopyTo(WaveformTo:4);

        //For intensity layer 0 , we want to mute the high frequencies a little, because when the Vioilin is played softly there is less energy being delivered to the instrument , and since high frequency evergy gets damped faster then this is effectively low-pass filtering.
        WaveSet.SetCurrentIntensityLayer(IntensityLayer:0);
        CurrentIntensityLayer.ShapeTheHarmonics(Direction:Up,HarmonicIDFrom:2,Slope:-3dB);

        //We sprinkle in some randomness into the Harmonic levels across the timbral landscape. The sample data was all at a single intensity but was copied across all intensities. So let's add just a little randomness.
        //This also means that all waveforms are noticeably different when we apply timbral modulation between them.
        WaveSet.AddHarmonicRandomness(10%);

        //We overblow the upper intensity layer (shifts harmonic levels to the right, setting the fundamental at a level of zero).
        //WaveSet.OverblowUpperIntensityLayer();

}
//Define a low-pass filter with a cut-off at 2KHz and a 12dB per octave slope.
```

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

Define.Patch
{
	General.SetEnvelopeGainController(Envelope:Amplitude,InitialLevel:0,GainCC:Velocity);
	General.SetEnvelopeGainController(Envelope:TimbreMorph,InitialLevel:0,GainCC:None);

	//We use the attack phase just for the breath noise. This makes sense since the sound has got to follow the breath. Judging the best amount of delay until the sound is by trial and error.
	ADSR.ConfigureSection(Section:Attack,Enabled:True,Duration:50,EndKSU:0%,EndKSL:0%,Sample:None,SampleMode:OneShot);
	//We use a tiny level of sample (one-shot) in the decay section that sounds like an initial burst of air through the lips. This is almost imperceptable but makes a difference.
	ADSR.ConfigureSection(Section:Decay,Enabled:True,Duration:1304,EndKSU:0%,EndKSL:0%,Sample:KeyUp,SampleMode:OneShot);
	ADSR.ConfigureSection(Section:Sustain,Enabled:True,Duration:20000,EndKSU:0%,EndKSL:0%,Sample:None,SampleMode:OneShot);
	ADSR.ConfigureSection(Section:Release,Enabled:True,Duration:400,EndKSU:0%,EndKSL:0%,Sample:None,SampleMode:OneShot);

	//===== GENERAL : Misc ==========
	//Add 1 cent of random detuning, because even the best player will have some inaccuracy.
	General.SetDetuning(DetuningType:Random,Detuning:1%);
	//We turn on Railsback Inharmonicity, because the recorder physics results in a certain amount of inharmonicity.
	General.SetHarmonicAlgorithm(HarmonicAlgorithm:RailsbackInharmonicity);
	General.SetScalingSplit(Note:c4);

	//Set to 'boundary' velocity curve, so that there is a sharp transition of intensity when the input velocity rises beyond a threshold (MIDI note velocity 91).
	General.SetVelocityCurve(Curve:Boundary);

	//The timbre can overload a little so reduce the patch gain
	General.SetPatchGain(Gain:80%);

	//===== GENERAL : Oscillators ==========
	//Adding another oscillator, slightly detuned enriches the sound. Increasing the detune above 2 can start to have dramatic phasing effect which is really interesting, but let's keep it more realistic but still there to enrich the sound.
	General.SetActiveOscillators(OscillatorCount:2);
	General.SetDetuningMode(DetuningMode:Cents);
	General.SetOscillatorDetuning(Oscillator:0,Detuning:0.00);
	General.SetOscillatorDetuning(Oscillator:1,Detuning:1.00);
```

```
    //===== GENERAL : Envelope control ==========
    General.SetEnvelopeGainController(Envelope:TremoloDepth,InitialLevel:0,GainCC:None);
    General.SetEnvelopeGainController(Envelope:NoiseGain,InitialLevel:0,GainCC:Velocity);
    General.SetEnvelopeGainController(Envelope:TimbreLFODepth,InitialLevel:0,GainCC:None);
    General.SetEnvelopeGainController(Envelope:SampleGain,InitialLevel:0,GainCC:Velocity);
    //Add a vey slight LPF on the noise
    General.SetEnvelopeGainController(Envelope:NoiseCutoffFrequency,InitialLevel:80,GainCC:None);
    General.SetEnvelopeGainController(Envelope:VibratoDepth,InitialLevel:0,GainCC:None);

    //===== GENERAL : LFOs ==========
    //Generally players use Tremolo, by varying the blown air pressure. around 3Hz sounds right.
    General.SetLFO(LFOType:Tremolo,Enabled:True,WaveType:Sine,Frequency:2.7,FrequencyCC:None,DepthCC:Modulation);
    //We modulate the Timbre at the same frequency as Tremolo, and because we've randomised the harmonic levels a bit actoss all waveforms, then this will vary the timbre a bit, all adding to richness of sound.
    General.SetLFO(LFOType:TimbreLFO,Enabled:True,WaveType:Sine,Frequency:2.7,FrequencyCC:None,DepthCC:Modulation);
    //The 'Fluctuating' vibrato option, coupled with the vibrato depth settings, emulate the 'wobbliness' of playing note pitches.
    //If the envelope depth setting is too high then it sounds like a ridiculously poor player on sustained notes - so it's been set at the threshold of 'poor', just to make it sound realistic.
    General.SetLFO(LFOType:Vibrato,Enabled:True,WaveType:Fluctuating,Frequency:1.00,FrequencyCC:None,DepthCC:None);

    //----- ENVELOPES : for ADSR section 'Attack' ----------
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:Amplitude,EnvelopeType:Linear,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:38,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:1.528%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:NoiseGain,EnvelopeType:Exponential,Target:2.0%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:10.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:NoiseCutoffFrequency,EnvelopeType:None,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:TimbreMorph,EnvelopeType:Linear,Target:25.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:125,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:SampleGain,EnvelopeType:Exponential,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:0.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:PitchShift,EnvelopeType:None,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:TremoloDepth,EnvelopeType:Exponential,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.250%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:VibratoDepth,EnvelopeType:Exponential,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.125%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Attack,Envelope:TimbreLFODepth,EnvelopeType:Exponential,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.125%,ExpMultKSU:0%,ExpMultKSL:0%);

    //----- ENVELOPES : for ADSR section 'Decay' ----------
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:Amplitude,EnvelopeType:Exponential,Target:100.0%,TargetKSU:0%,TargetKSL:0%,LinearDelta:13,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:0.278%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:NoiseGain,EnvelopeType:Exponential,Target:1.0%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:NoiseCutoffFrequency,EnvelopeType:None,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:TimbreMorph,EnvelopeType:Linear,Target:50.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:724,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:SampleGain,EnvelopeType:Exponential,Target:0.35%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:25.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:PitchShift,EnvelopeType:None,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:TremoloDepth,EnvelopeType:Exponential,Target:70.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.250%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:VibratoDepth,EnvelopeType:Exponential,Target:50.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.125%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Decay,Envelope:TimbreLFODepth,EnvelopeType:Exponential,Target:100.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.125%,ExpMultKSU:0%,ExpMultKSL:0%);

    //----- ENVELOPES : for ADSR section 'Sustain' ----------
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:Amplitude,EnvelopeType:Exponential,Target:70.0%,TargetKSU:0%,TargetKSL:0%,LinearDelta:66,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:0.1%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:NoiseGain,EnvelopeType:Exponential,Target:1.0%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:2.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:NoiseCutoffFrequency,EnvelopeType:None,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:TimbreMorph,EnvelopeType:Linear,Target:50.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:1,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:SampleGain,EnvelopeType:Exponential,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:0.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:PitchShift,EnvelopeType:None,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:TremoloDepth,EnvelopeType:Exponential,Target:70.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.125%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:VibratoDepth,EnvelopeType:Exponential,Target:50.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.125%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Sustain,Envelope:TimbreLFODepth,EnvelopeType:Exponential,Target:70.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:1.000%,ExpMultKSU:0%,ExpMultKSL:0%);

    //----- ENVELOPES : for ADSR section 'Release' ----------
    //In the release we hold the vibrato and tremolo values constant.
    ADSR.ConfigureEnvelope(Section:Release,Envelope:Amplitude,EnvelopeType:Exponential,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:1,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:1.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:NoiseGain,EnvelopeType:Exponential,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:0.1%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:NoiseCutoffFrequency,EnvelopeType:None,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:TimbreMorph,EnvelopeType:Linear,Target:75.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:533,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:SampleGain,EnvelopeType:Exponential,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:0.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:PitchShift,EnvelopeType:None,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:TremoloDepth,EnvelopeType:Linear,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:10.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:VibratoDepth,EnvelopeType:Linear,Target:0.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:30.000%,ExpMultKSU:0%,ExpMultKSL:0%);
    ADSR.ConfigureEnvelope(Section:Release,Envelope:TimbreLFODepth,EnvelopeType:Linear,Target:70.00%,TargetKSU:0%,TargetKSL:0%,LinearDelta:0,LinearDeltaKSU:0%,LinearDeltaKSL:0%,ExpMult:3.125%,ExpMultKSU:0%,ExpMultKSL:0%);
}
```