SONICCHARGE

# BitSpeek Re

version 1.0

# Table of Contents

Introduction....3
User Interface Front....4
User Interface Back....5
Using Bitspeek with Stereo Signals....7
Bitspeek Versus Vocoders....8
Example Combinators....8
Credits and Contacts....12
Copyrights And Trademarks....12

# INTRODUCTION

Bitspeek is a real-time pitch-excited linear prediction codec effect. Right now you are probably thinking, "oh, another one of those"? Or perhaps not. Chances are that you have never heard about "linear prediction", although most of us use it daily when we talk on our cell phones. Linear prediction coding is a voice compression technology that appeared in commercial products in the seventies and was implemented in some well-known speaking toys of the early eighties.

We have applied this technology to create an audio effect that analyzes audio, extracts a number of parameters (including pitch, volume and formant data) and then resynthesizes the audio using a simple oscillator, noise and filter architecture.

A number of playback parameters can adjust the pitch and tonal quality of the sound. It is possible to play the pitch with MIDI notes and you can create tempo-synchronized "formant freezing effects". Despite having only a few simple controls, this box can produce a broad range of sounds from cheap speaking toys to high-end vocoder and talkbox effects.

# User Interface Front


## RATE *(kHz)*

Bitspeek performs its calculations at a designated fixed sample-rate, regardless of the sample-rate your project is running at. The possible settings are 8 kHz, 11 kHz, 22 kHz and 44 -> 48 kHz *(the last mode will actually adjust to the project sample-rate and select a rate between 44 and 48 kHz)*. Notice that Bitspeek is still compatible with any sample-rate by performing automatic sample-rate conversion.

RATE affects several other internal parameters in the DSP algorithms and changing it alters the sound dramatically. *(Notice that the 44 kHz mode will require more CPU compared to the other modes. In many cases, the 22 kHz mode works just as well.)*

## FRAME RATE

The audio signal is analyzed and processed in blocks called "frames". For each frame, Bitspeek estimates the pitch, volume, stereo image and formants of the incoming audio, as well as the balance between "voiced" audio *(e.g vowels)* and "voiceless" audio *(e.g., the noise in a consonant)*.

By lowering the frame rate, the analysis will be performed more rarely and you will achieve a cheaper toy-like sound. You can also "freeze" the audio by dragging FRAME RATE all the way down to 0. The parameter range is 0 to 80 Hz *(if SYNC is off)* and higher rates requires more CPU than lower rates.

## SYNC

Enable SYNC to make Bitspeek "freeze" frames in sync with the tempo of your *music*. When SYNC is enabled, you may select various time synchronized rates *(1/8, 1/16 etc)* with the FRAME RATE slider instead of selecting a rate in Hz.

## KBD MODE

Enable this mode to control the pitch and envelope of the effect with your MIDI keyboard. When enabled, Bitspeek will play only when it receives notes and it will transpose the pitch according to the notes it receives.

Start by creating a sequencer track for Bitspeek *(e.g., right-click the device and choose "Create Track for Bitspeek")*. Turn down the TRACKING parameter to zero to achieve a vocoder / auto-tune like sound.

If you set the FRAME RATE to zero *(with SYNC disabled)*, Bitspeek will "freeze" the formants on note on, allowing you to create interesting "stroboscopic" audio effects. Finally, Bitspeek supports Pitch Bend *(one octave up and down)* and the Sustain Pedal can also be used to "freeze" frames while playing.

### PITCH
You can transpose the generated audio by -36 to +36 semitones *(-3 to +3 octaves)*.

### FINE
Fine pitch adjustment from -100 to +100 cent.

### TRACKING
Determines the amount by which the synthesized audio follows the pitch of the source signal, from 0% to 200%. At 0%, the pitch will stay fixed and produce a robotic vocoder-like quality. At 100%, the processed audio will follow the pitch intonation of the original audio as closely as possible. *(Sometimes though, the tracking detects the wrong octave, especially on source material with extremely low pitch.)*

Notice that PITCH, FINE and TRACKING also affects the CV outputs on the backside *(NOTE and PITCH)*.

### DETUNE
There is a second oscillator which can be used to achieve a fat detuned sound or for chord-like effects. The second oscillator is transposed from the first by +0 to +1200 cents, representing a range of up to one octave.

### NOISE (adjust)
This parameter adjusts the balance of "voiced" versus "voiceless" sound. At the default setting +/- 0%, Bitspeek attempts to follow the balance of the source signal, so that "voiced" sounds *(e.g. vowels)* produce distinct tones while "voiceless" sounds *(e.g. consonants)* produce noise.

By turning NOISE all the way down to -100%, all noise will be removed from the output audio. By turning NOISE up to +100%, the output audio will consist only of filtered noise *(sounding like a loud whisper)*.

## User Interface Back


### AUDIO IN
Connect AUDIO IN to the source that should be analyzed and recreated. Bitspeek works with stereo as well as mono sources.

### EXT IN
Use "external input" to apply the frequency spectrum of the incoming audio onto other signals, e.g. to make any synthesizer in Reason speak. When you connect

EXT IN, the internal synthesizer in Bitspeek will be disabled and the external signal will be processed by the formant filter instead. There is a built-in pre-emphasis stage that boosts higher frequencies to prevent the result from becoming too "muddy".

### AUDIO OUT

The output signal. If you have connected both input channels *(or external input channels)* you would normally want to connect both outputs as well. Bitspeek will not automatically mix or split signals from stereo to mono or vice versa.

### GATE IN

The gate input is most often used together with NOTE IN to control Bitspeek from a CV source such as the Matrix Pattern Sequencer. GATE IN and NOTE IN works both with KBD MODE disabled and enabled. When KBD MODE is enabled, Bitspeek will only produce audio when the gate CV signal is high.

Regardless if NOTE IN is connected or not, the gate input can be used to force triggering new "frames". Set FRAME RATE to 0 to use this feature.

### NOTE IN

This CV input lets you control the pitch of the built-in synthesizer *(and also of the NOTE and PITCH CV outputs)*. Just as with KBD MODE you normally want to turn TRACKING down to 0% when connecting NOTE IN.

### PITCH IN

PITCH IN is a pitch adjustment input with scalable range. Unlike NOTE IN, this input is bipolar and it is not "quantized" to semitones. You can use it to add modulation to the synthesized pitch or to hook up one Bitspeek to another to follow the exact same pitch.

### GATE OUT

This output works both as a gate signal *(to trigger other synths, etc.)* and as an envelope follower. The output is logarithmic with respect to the input amplitude *(unlike the VOICE and NOISE outs, which are linear)*. In other words this output works as a dB measurement and the range is -36 to 12 dB. When connected to the gate input of a synth in Reason, the synth will trigger a voice when the signal raises above approximately -35.5 dB and the device will release the voice when it drops below again.

### NOTE OUT

This output will transmit the pitch of the analyzed signal. The output is scaled, offset and quantized to the standard CV note range in Reason. This makes it easy to wire a synth to play along with Bitspeek. Just connect the GATE and NOTE out to the synth and set it to "monophonic legato" mode. Route the audio of the synth back into EXT IN on Bitspeek and you have rolled your own flavor of robotic speech.

### PITCH OUT

Unlike NOTE OUT, PITCH OUT is bipolar and not quantized to semitones. It provides an alternative to connecting NOTE OUT if you need more exact pitch tracking. However, it requires a slightly more complicated setup, depending on implementation details of the synth you are connecting. E.g. Thor uses a different range for CV pitch modulation than SubTractor and Malström.

When PITCH OUT is connected, the NOTE OUT signal will be fixed at middle C. This is a convenience feature, allowing you to connect both CVs and letting PITCH OUT be responsible for pitch tracking and NOTE OUT simply provide the base note for correct tuning.

### STEREO : PAN OUT

The analyzed panning position of the source signal. Bipolar, so 0 is center, -1 is full left and 1 is full right.

### STEREO : WIDTH OUT

The analyzed stereo width of the source signal, in other words how much the left and right input channels correlate. An output of 0 means 100% correlation *(i.e. monophonic)* and an output of 1 means completely uncorrelated.

### VOLUME : VOICE OUT

The linear amplitude of the "voiced" part of the source signal. You can use this CV to control the amplitude of external oscillators fed back into EXT IN.

### VOLUME : NOISE OUT

The linear amplitude of the "voiceless" *(noisy)* part of the source signal. You can use this CV to control the amplitude of a noise generator fed back into EXT IN.

## Using Bitspeek with Stereo Signals

The Rack Extension version of Bitspeek can be used with stereo signals. It will analyze the stereo image of the source signal and extract two parameters: panning and stereo width *(corresponding to the amplitude balance and the correlation of left and right signals)*. Bitspeek will then attempt to mimic the stereo image with the built-in synthesizer.

The oscillator is only monophonic *(but panned)* while the noise is stereophonic and copies the stereo width of the source signal. This solution opens up for some interesting pseudo-reverb effects.

# Bitspeek Versus Vocoders

Vocoder is short for "voice encoder" and technically that means any device that is capable of breaking down a voice signal into a set of parameters from which you can reconstruct the voice intelligibly. With this terminology, Bitspeek may be called an "LPC vocoder". *(LPC is short for "linear prediction coding".)*

However, in musical contexts, the vocoder we all know is a device consisting of a filter bank that is controlled from a modulation source *(e.g. speech)*. This filter bank is fed with an arbitrary carrier signal *(typically from a synthesizer)* that is played independently of the modulation source.

This is very different from how Bitspeek works. The formant filter in Bitspeek is not implemented as a bank of bandpass filters. Instead, the analysis is divided into time frames *(typically around 10ms each)*. For each frame the algorithm quickly constructs a resonant filter that closely represents the formants of the input speech.

In other words, whereas conventional vocoders works with a number of bandpass filters at stationary frequencies, Bitspeek creates animated filters whose peaks are precisely positioned in the spectrum. On the other hand, the signal is chopped up in discrete time frames, whereas the conventional vocoder varies the formants smoothly over time.

# Example Combinators

Bitspeek for Rack Extensions ships with some example Combinator patches. They were created to be inspirational starting points for further exploration of the interconnectivity possibilities in Reason. Here is the run down with screenshots and explanations.

## ToyTalk (plain)

This is a plain single Bitspeek only, mainly provided as a reference point. It is configured to produce a sound reminiscent of a certain early 80's toy.

## SubSpeek

One Bitspeek controls the pitch and the volume of a SubTractor, which in turn is routed back into Bitspeek for filtering. In effect, the SubTractor will replace the built-in synth in Bitspeek, enabling completely new timbres.

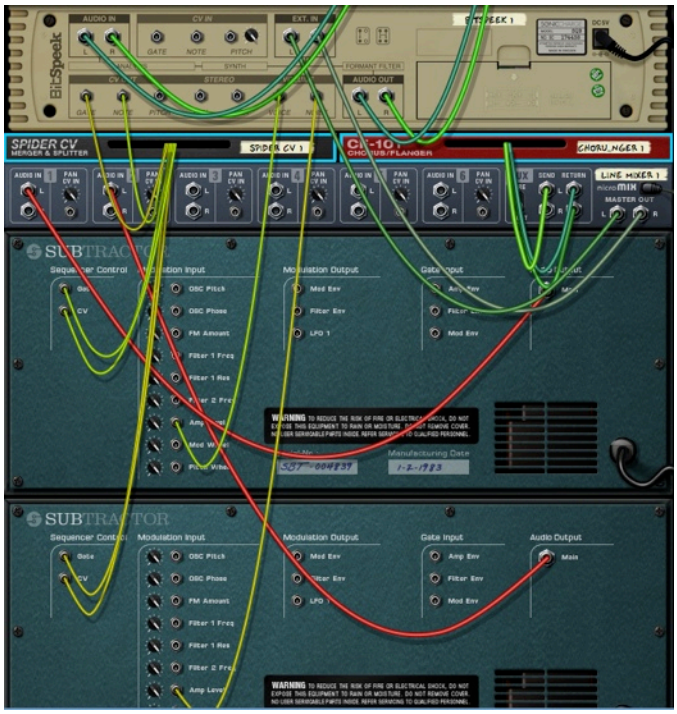

Notice how we have connected both NOTE and PITCH CV OUT to achieve exact pitch tracking. VOICE VOLUME OUT to "Amp Level" will make the SubTractor play voiced sounds only *(and not noise)*.

## FMTalk

This wacky lo-fi patch is similar to SubSpeek but it does not connect PITCH CV OUT. Instead it relies on NOTE CV only *(which is "quantized" to semitones)*.

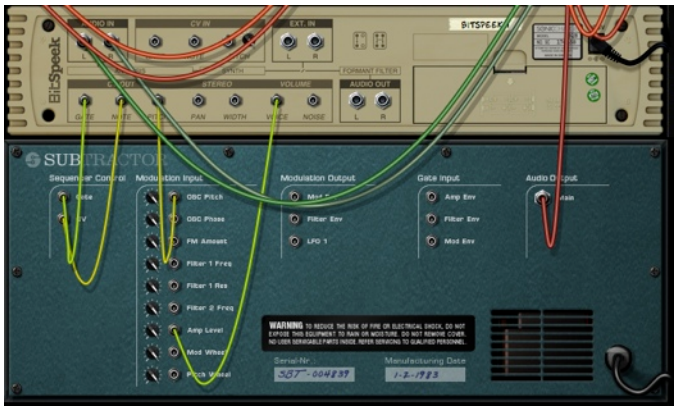
The SubTractor patch is set to "Legato" and has a little "Portamento" to make pitch changes smoother. "Amp Level" is scaled less than 100% which produces a slightly compressed result.

## ChoruSpeek



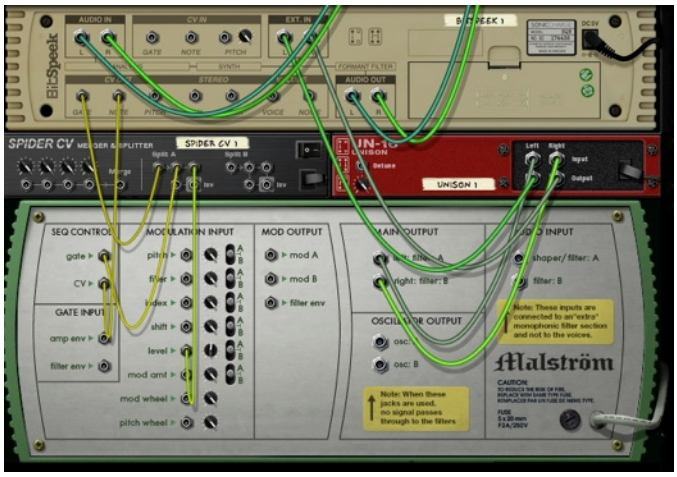
Another Bitspeek ↔ SubTractor Combinator, this time with two SubTractors and a little chorusing before returning the audio to Bitspeek. One SubTractor provides the voiced sound and the other provides noise.

## SubLayer



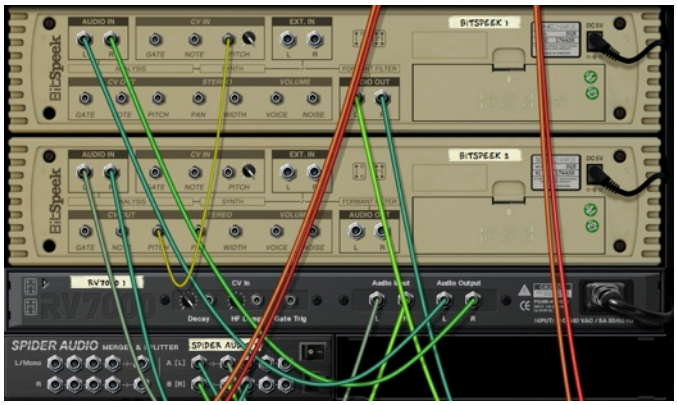
Here, Bitspeek is simply controlling SubTractor to make it play along with the audio you feed into the Combinator. Unlike the Combinator patches described above, the SubTractor audio is not fed back into Bitspeek.

## MalSpeek



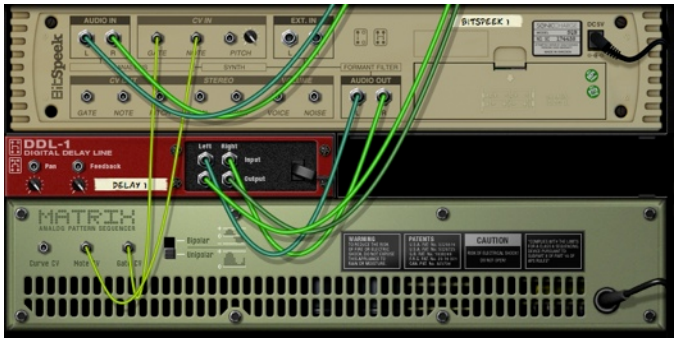

This Combinator uses Malström instead of SubTractor. Malström's "level" CV input uses a logarithmic curve instead of the linear one found on SubTractor. Luckily we have a logarithmic volume output on Bitspeek too and it is called GATE. A Spider does the job of feeding the GATE both to "gate" and "level" on Malström.

## BitFreeze




With the default Combinator settings, Bitspeek is using SYNC mode with a pretty low FRAME RATE and it is set to full NOISE. Reverb is added to the Bitspeek input to create wide sample-and-hold like stereo noises. For fun, if you turn NOISE down, another instance of Bitspeek provides pitch tracking for the first instance. With this technique, the pitch will not be "frozen" with the slow frame rate as it normally would. This makes the sound more harmonic when mixing in the source signal.

## TranSpeek




A simple setup where a Matrix Pattern Sequencer is triggering notes in Bitspeek, using it almost like a monophonic synth with a filter that is controlled by the audio input to Bitspeek. A little delay is added for effect.

## TalkingDrums


This time we are feeding drums into EXT IN to filter them with the formants of the audio that is entering AUDIO IN. With the default Combinator settings, the FRAME RATE is set to SYNC on 16th notes, producing a sample-and-hold like quality. Subtle reverb and compression is applied before feeding the signal to Bitspeek.

## NoiseWash


Here is an interesting concept. The Combinator audio input is split with a Spider and fed both to AUDIO IN and EXT IN, thus Bitspeek will apply the formants of the source audio onto itself. The net effect is that spectral peaks will be boosted and the noise floor will be attenuated, similar to how spectral gating in noise reduction algorithms work. A compressor keeps the output level in check.

## SpeekBox (midi)


The final Combinator patch needs to be played with midi. It attempts to emulate a "talkbox" effect by sending a SubTractor to EXT IN.

We disregard the volume and the pitch of the speech signal *(going to AUDIO IN)*. We are just after the glorious formants.

To avoid undesirable glitches when the speech signal dips low, a noise gate contraption has been devised using Thor and CV from an MClass Compressor. All it basically does is mute the speech signal when it drops below a certain threshold. Bitspeek will not attempt to extract formants on silent input, thus effectively freezing the last audible content.

So, if you set the "Threshold" knob on the Combinator to an appropriate level you should be able to catch your breath without interrupting audio while you crank out those funky talkbox leads.

# Credits and Contacts

Sonic Charge Bitspeek RE v1.0.0 *(2012)*

Created by:

Magnus Lidström

Graphical design and additional development:

Fredrik Lidström

Sonic Charge website:

http://soniccharge.com

# Copyrights And Trademarks



The contractor / manufacturer for Sonic Charge Bitspeek is:

Magnus Lidström
Mosebacke Torg 16 A
S-116 20 Stockholm
Sweden