µBraids SE

Eurorack Module

TD–UBSE–C

Designed by Michael & Daniel Gilbert of *Tall Dog Electronics* in Western Massachusetts
tall-dog.com | **ubraids-se.com**

Based on the *microBraids* (aka *µBraids*) module designed by *Magpie Modular* and *Neutron Sound*
magpiemodular.com
neutron-sound.com

Based on the *Braids* module designed by Olivier Gillet of *Mutable Instruments*
mutable-instruments.net

Manufactured by *CircuitHub* via *Worthington Assembly* in South Deerfield, Massachusetts
worthingtonassembly.com
circuithub.com

Source materials are available on *GitHub*
github.com/loglow/uBraids_SE

Contents

About

µBraids SE is a voltage-controlled digital oscillator/ sound source in 8HP with a wide range of creative and useful sound generation algorithms (aka models).

This redesigned *µBraids SE* (Special Edition) is a new revision of *µBraids* that updates and simplifies the component layout and has a new panel design. It utilizes a custom miniature segmented LED module, eliminating the need for a 2nd microprocessor and its supporting circuitry. *µBraids SE* remains fully compatible with future upstream *Braids* firmware upgrades and any other alternate firmwares that are compliant with the standard *Braids* design.

Included Parts

µBraids SE includes a standard 16-pin to 10-pin Eurorack power cable and two sets of mounting screws for racks/enclosures with either M2.5 or M3 threads.

History

The original *Braids* (**mutable-instruments.net/modules/braids**) was designed by Olivier Gillet of *Mutable Instruments* (**mutable-instruments.net/about**) under a CC BY-SA 3.0 (**creativecommons.org/licenses/by-sa/3.0**) license. *Braids* is a macro-oscillator with multiple intricate digital synthesis algorithms. Each algorithm is controlled by two continuously variable parameters, both of which are voltage controllable. It is available as a 16HP Eurorack module.

Source: **github.com/pichenettes/eurorack**

The original *microBraids* (aka *µBraids*) (**store.magpiemodular.com/products/microbraids-pcb**) is a full-functionality adaptation of *Braids* designed by *Magpie Modular* (**magpiemodular.com**) and *Neutron Sound* (**neutron-sound.com**) and is available only as a bare PCB (printed circuit board). It is 8HP (exactly half the original width) and features an OLED display driven by a 2nd processor. Nothing else has been altered component-wise so any future firmware upgrades or alternate firmware such as *Bees-in-the-Trees* (**timchurches.github.io/Mutated-Mutables**) will work fine.

Source: **github.com/MagpieModular/microBraids**

Warranty

This product is covered under warranty for one year following the date of purchase as indicated on the original sales receipt. This warranty covers any defect in the manufacturing of this product. This warranty does not cover any damage or malfunction caused by incorrect use—such as, but not limited to, power cables connected backwards, excessive voltage levels, exposure to extreme temperature or moisture levels, physical damage due to impact, or any aftermarket physical or electrical modifications or repairs.

This warranty covers either repair or replacement, at our sole discretion. This repair or replacement is subject to verification of the defect or malfunction and proof of purchase as confirmed by showing the model number on an original dated sales receipt. Shipping and handling fees are to be paid for by the customer.

Please contact **support@tall-dog.com** for a return authorization prior to shipping anything to us. Please understand that we will not be able to service units under warranty that have been modified or previously repaired by the customer or a third party.

Inputs & Outputs

TRIG — This trigger input serves three purposes:

1. Certain physical models need to be triggered by a pulse on this input to generate a sound.
2. The other models will treat the trigger as a reset signal, bringing the phase of the oscillator(s) to 0°.
3. This input can also be used to trigger an internal AD envelope applied to the parameters of your choice, to create sound animation and attacks without an external envelope module.

1V/OCT — 1V/octave frequency CV input.

FM — Frequency modulation CV input. The scale and polarity of this signal is set by the FM attenuverter.

TIMBRE and **COLOR** — Control voltages for the Timbre and Color parameters. A value of 0V corresponds to the minimum position of the knob. A value of +5V corresponds to the maximum position of the knob. This CV is offset by the current position of the knob.

OUT — Signal output. Loudness is model-dependent. For example, a pure sine wave is always at maximum amplitude; while a ring-modulated sine-wave will have peaks and valleys due to amplitude modulation, and will thus sound quieter.

Models

CSAW

This model is inspired by a quirk/defect of the *Yamaha CS80* sawtooth wave shape, consisting of a fixed-width “notch” after the raising edge. The width of the notch can be controlled by **TIMBRE** and its depth and polarity can be controlled by **COLOR** to produce phasing effects.

This model produces the classic waveform trajectory from triangle to sawtooth to square to pulse found in synthesizers such as the *RSF Kobol* or the *Moog Voyager*. **TIMBRE** sweeps through the waveforms. **COLOR** morphs from several tonal characters by increasingly removing the high-frequencies with a 1-pole filter, and recreating them with a waveshaper.

This model blends a sawtooth wave with dephasing control, with a square wave with PWM. **TIMBRE** controls the dephasing amount or pulse width, and **COLOR** morphs the waveshape from sawtooth to square.

Models *Cont'd*

FOLD

This model is built with sine and triangle oscillators sent into a wavefolder. **TIMBRE** controls the wavefolder strength, and **COLOR** controls the balance between the sine and triangle signals sent to it.

⊥⊥⊥⊥

This digital synthesis algorithm generates a smooth sequence of waveforms, transitioning from a sine wave to a Dirac comb, as controlled by **TIMBRE**. The intermediary steps are reminiscent of a single formant. Two such waveshapes are blended together, with the detuning amount controlled by **COLOR**.

SUB⊓ SUB/|

Sub oscillator modes with a square or saw waveform. These two models were added in a firmware update after the original *Braids* module was first released.

Models *Cont’d*

SYN⊓ SYN/|

These models synthesize the classic 2-oscillator hard-sync patch, with both oscillators emitting square or saw waves. The main oscillator frequency controls the master frequency. The interval between master and slave is controlled by **TIMBRE**. **COLOR** controls the balance between the two oscillators.

/|/|X3 ⊓_X3 /\X3 SIX3

Three sawtooth (or square, triangle, sine) oscillators which can be individually tuned. **COLOR** and **TIMBRE** control the relative frequency of the second and third oscillator with respect to the main oscillator. These two controls are quantized to “snap” on musical intervals like octaves or fifths.

RING

Three sine wave oscillators are ring-modulated together, and colored by a waveshaper. The main oscillator frequency controls the frequency of the first sine wave, and **TIMBRE** and **COLOR** control the relative frequency of the second and third sine waves.

Models *Cont’d*

This model simulates a swarm of 7 sawtooth waves. **TIMBRE** controls their detuning, and **COLOR** applies a high-pass filter to the resulting sound.

This model generates a sawtooth waveform, and sends it into a comb filter (tuned delay line). The frequency of the delay line tracks the frequency of the sawtooth oscillator, with a transposition controlled by the **TIMBRE** knob. **COLOR** selects the feedback amount and polarity: at 12 o’clock, no feedback is applied. From 12 o’clock to 5 o’clock, positive feedback is increasingly applied. From 12 o’clock to 7 o’clock, negative feedback is progressively applied.

TOY*

This model traverses a space of timbres typical of (circuit-bent) electronic musical toys. **TIMBRE** simulates an alteration of the toy’s clock rate, while **COLOR** creates glitches or short-circuits on a converter or memory chip’s data lines.

Models *Cont'd*

ZLPF ZPKF ZBPF ZHPF

This family of models directly synthesize, in the time-domain, the response of a low-pass, peaking, band-pass or high-pass filter excited by classic analog waveforms. Rather than synthesizing the waveform and filtering it (which is what a VA synthesizer would do), this approach directly aims at building the filtered waveshape from scratch. This technique has been used in the *Casio CZ* or the *Roland D* series, but is extended here to cover different filter types and waveshapes. **TIMBRE** controls the cutoff frequency of the filter. **COLOR** continuously modifies the waveshape, from saw to square to triangle.

VOSM

This model uses a combination of 3 oscillators arranged in a clever ring-modulation/hard-sync patch to emulate formant synthesis—a technique named VOSIM and described by Kaegi and Tempelaars. **COLOR** and **TIMBRE** control the relative frequencies of the two formants.

Models *Cont'd*

VOWL VFOF

Both models synthesize vowel sounds. VOWL is a faithful recreation of early computer speech synthesis programs. VFOF uses a simplified version of Rodet's FOF synthesis technique. Both have the same control layout: **TIMBRE** controls the vowel, morphing between a, e, i, o, u. **COLOR** shifts the formants in frequency. Main oscillator frequency and **COLOR** can be used altogether to simulate age and gender transformations.

HARM

This model uses additive synthesis, by summing 12 sine harmonics. **COLOR** modifies the distribution of the amplitudes of each harmonic, around a central frequency set by **TIMBRE**.

FM__ FBFM WTFM

Three flavors of 2-operator phase-modulation synthesis. **TIMBRE** controls the modulation amount. **COLOR** controls the relative frequency interval between modulator and carrier. FM__ is a well-behaved implementation. FBFM uses feedback from the carrier to itself to produce harsher tones. WTFM uses two feedback paths, from carrier to modulator and carrier to itself to achieve droning, unstable tones.

Models *Cont'd*

PLUK

Raw plucked string synthesis. **TIMBRE** controls the damping, **COLOR** the plucking position. A trigger or gate signal is required.

BOWD

Bowed string modeling. **TIMBRE** controls the friction level, **COLOR** the bowing position. A trigger or gate signal is required. Note that this model does not include a body filter—which would be necessary to simulate an actual string instrument.

BLOW FLUT

Reed or flute instrument model. **TIMBRE** controls the air pressure, **COLOR** the geometry of the instrument. A trigger or gate signal is required. Note that this model does not include a filter—which would be necessary to simulate an actual wind instrument.

Models *Cont'd*

BELL

This model established by Risset uses additive synthesis to recreate the tone of a bell. **TIMBRE** controls the damping of the sound; and **COLOR** the inharmonicity of the sound. A trigger or gate signal is required.

DRUM

This variant of the BELL model uses different parameters (partials frequencies and amplitudes) to generate a sound reminiscent of a metallic drum. **TIMBRE** controls the damping and **COLOR** the brightness. A trigger or gate signal is required.

KICK

This model is a simulation of the *TR-808* bass drum circuit. **TIMBRE** controls the decay time, while **COLOR** controls the brightness (tone) of the sound. The main oscillator frequency controls the tuning of the bridged-T filter. A trigger or gate signal is required.

Models *Cont'd*

CYMB

Raw material for cymbal sound synthesis, as inspired by the *TR-808*. **COLOR** controls the balance between a droning sum of square waves and noise. **TIMBRE** controls the cutoff of a band-pass filter applied on the resulting signal. A trigger or gate signal is required.

SNAR

This model is a simulation of the *TR-808* snare drum. **TIMBRE** controls the balance between the two modes of the resonator (tone) and **COLOR** controls the amount of noise (snare). A trigger or gate signal is required.

WTBL

This model is a classic implementation of wavetable synthesis. **TIMBRE** sweeps the wavetable, and **COLOR** selects one of the 20 wavetables to play with. The waveforms are interpolated when traveling through a wavetable, but not when switching from one table to another.

Models *Cont’d*

WMAP

This model is a two-dimensional implementation of wavetable synthesis. 256 waveforms have been laid out in a 16×16 grid, so that adjacent waveforms are similar sounding. The **TIMBRE** parameter scans the table in the X direction, and the **COLOR** parameter scans the table in the Y direction, with smooth interpolation across the two directions.

WLIN

This model allows one dimensional scanning through the entirety of *Braids’* wavetables. **TIMBRE** moves through the waves, while **COLOR** selects the interpolation method. When **COLOR** is at 7 o’clock, no interpolation is applied. When **COLOR** is at 10 o‘clock, interpolation is applied between samples, but not between waves. When **COLOR** is at 12 o’clock, interpolation is always applied. When **COLOR** goes past 12 o’clock, interpolation is applied between waves, but the resolution of the playback resolution is decreased.

WTX4

This mode is a 4-voice variant of WLIN. **TIMBRE** morphs through a small selection of 16 waves. **COLOR** selects the harmonic structures between the 4 voices from a predefined set of chords. When **COLOR** is at 7 o'clock, all voices are playing the same note with a variable amount of detuning, creating a thick chorus effect.

NOIS

This model filters white noise with a state-variable filter. The main oscillator frequency controls the cutoff frequency of the filter. **TIMBRE** controls the resonance of the filter. **COLOR** performs a crossfade between the low-pass and high-pass outputs of the filter.

TWNQ

This dual-peak model generates white noise and process it with two band-pass filters (resonators). **TIMBRE** controls the Q factor of the filters, and **COLOR** changes their spacing. The frequency of both filters each track the main frequency.

CLKN

This model generates random samples at a given rate, determined by the main pitch control. **TIMBRE** controls the periodicity of the generator (up to a 2 samples cycle), and **COLOR** its quantization level (from 2 distinct values to 32 distinct values).

CLOU PRTC

These granular synthesis models create natural textures by mixing short grains of windowed sine waves (CLOU) or short, decaying pings (PRTC). The frequency of the grains is controlled by the main frequency control, but is randomized by an amount proportional to the **COLOR** control. **TIMBRE** controls the density and overlap of the grains.

QPSK

This model generates—in the audio frequency range—the kind of modulated signals used in digital telecommunication systems. The main oscillator frequency is the carrier frequency. The bit-rate is controlled by the **TIMBRE** knob. The **COLOR** knob sets an 8-bit value which is modulated into the carrier using QPSK modulation. A 16-byte synchronization frame is sent on every trigger/gate, or every 256 data bytes.

Options

META

Allows the synthesis model to be selected by the **FM** CV. When this mode is active, frequency modulation through the **FM** CV input is no longer possible—but is replaced by CV-controlled model selection.

This option is great for creating sequences featuring the different synthesis models. Keep in mind that discontinuities or glitches might be heard when switching from one model to the next!

The encoder can still be used to scroll through synthesis models; and the CV applied to the **FM** input allows you to scroll forward (positive voltage) or backwards (negative voltage) through the list.

BITS

Selects the bit-depth of the data sent to DAC.

RATE

Selects the refresh rate of the DAC. Note that a handful of models are internally rendered at 48kHz (instead of 96kHz); the difference between 48kHz and 96kHz might be non-existent for the most complex models. Note also that conversely, to reduce aliasing, the simplest models are rendered internally at 192kHz or 384kHz.

TSRC

Selects a trigger source:

- EXT. uses the gate/trigger jack.
- AUTO additionally tracks changes in the **1V/OCT** frequency input larger than a semitone and generates a trigger on each of these. This allows, for example, the physical models or the internal AD generator to be controlled by a note sequencer which does not provide gate outputs.

TDLY

Applies a delay between the moment the trigger is received and the moment the note is sounded on the physical models. Some CV-gate converters or sequencers sometimes have slow settling times, or have short timing errors between the refresh of their analog and digital outputs. Delaying the processing of the trigger allows the physical model to sample the accurate CV rather than a fluctuating one which can cause unwanted glitches or portamento-like effects at note onsets.

NATT NDEC

Select the attack and decay time of the internal AD envelope generator.

\FM \TIM \COL \VCA

Control the amount of modulation from the internal AD envelope generator to the **FM**, **TIMBRE**, **COLOR** and output amplitude parameters. When all these settings are null, the **TRIG** input works as a sync/reset input.

RANG

Chooses the range of the **COARSE** knob:

- EXT. adjusts the range to ±4 octaves around the note received on the **1V/OCT** input. Because of this, when no frequency CV signal is sent to the module the coarse button will have a bias towards low frequencies, which might not always be desirable. This would be the equivalent to sending a CV of 0V—corresponding to a very low note!
- FREE adjusts the range to ±4 octaves centered around C3 (261.5 Hz). This setting is recommended when the module is used with no external signal on the **1V/OCT** CV input.
- XTND provides a larger (extended) frequency range, but disables accurate **1V/OCT** scaling as a side effect.
- 440 locks the oscillator frequency to 440 Hz exactly—helpful for tuning another VCO.

Options *Cont'd*

OCTV

This option is a transposition (by octave) switch.

QNTZ

Applies quantizing to the incoming **1V/OCT** control voltage. The frequency can be quantized to semitones, or to one of the many available scales, or disabled.

ROOT

Selects the root note upon which the quantizer's scale is built.

FLAT

Applies detuning in the lower and higher frequencies, to recreate some of the tuning imperfections of VCOs.

Options *Cont'd*

DRFT

Recreates the drifting of a poorly designed VCO.

SIGN

Applies grungy glitch/waveform imperfections to the output signal. The exact behavior of this option is unique to each module built.

BRIG

Adjusts the screen brightness.

Calibration

Follow this procedure to compensate for any inaccuracies in your CV sources, or if your module has lost its 1V/octave calibration settings following a reset or the installation of alternate firmware.

1. Disconnect any signal from the **FM** input.
2. Connect the note CV output of a well-calibrated keyboard interface or MIDI-CV converter to the **1V/OCT** input.
3. Move the **COARSE** and **FINE** knobs to their 12 o'clock positions.
4. Select CAL ı from the options list.
5. Click and hold the encoder for 1 second. The screen will display > C2 ı
6. Send a voltage of 1V to the CV input.
7. Click the encoder once. The screen will display > C4 ı
8. Send a voltage of 3V to the CV input.
9. Click the encoder once to finish calibration.

This software calibration procedure can also be used to achieve compatibility with the 1.2V/octave standard.

You can also shift the range of the **COARSE** knob up or down by performing this calibration procedure with any pair of notes that are 2 octaves apart, and with the **COARSE** knob set to any starting position.

Extras

Following CALi in the menu is a screen showing a visual representation of the internal ADC readings for the **TIMBRE**, **COLOR**, **1V/OCT** and **FM** inputs. This page is helpful for visualizing the polarity and range of incoming CV signals.

The next option shows a scrolling line of text. **TIMBRE** controls the scrolling and a gate/trigger can be used to scroll the text left by one column.

To edit the text:

1. Press the encoder for more than 1 second.
2. Rotate the encoder to select the first character.
3. Click to move to the next character and continue editing.
4. Once the line of text has been composed, select the last character (all segments lit) to confirm.

At any time, you can also hold the encoder to leave the edit mode.

Firmware Warnings

Before starting the firmware update procedure, please double-check the following:

- Make sure that no additional sound (such as email notification sounds, background music, etc.) from your computer will be played during the procedure.
- Make sure that your speakers/monitors are muted or not connected to your audio interface—the noises emitted during the procedure are aggressive and can harm your hearing.
- On non-studio audio equipment (for example the line output from a desktop computer), you might have to turn up the volume to the maximum.

Firmware Update

The module comes with version 1.9 of the *Braids* firmware installed. Perform the following steps to update the module's firmware via a provided audio file:

1. Unplug all CV inputs/outputs from the module.
2. Connect the output of your audio interface/sound card to the **FM** input.
3. Set the **FINE** knob to 12 o'clock.
4. Set the **FM** attenuverter to 5 o'clock.
5. Power on your modular system while depressing the encoder. The screen will show _RDY, with a "rotating snake" pattern on the first character.
6. When you are all set, play the firmware update file into the module. The display shows the number of data packets received. The firmware contains between 90 and 112 packets, and the unit reboots after the last packet has been received.

In case the signal level is too weak, the unit will display @SYN. Try adjusting the position of the FM attenuverter, click the encoder and retry from the start of the update file.

The unit displays @CRC if a data packet is corrupted. In this case, retry the procedure from another computer/ audio interface, and make sure that no piece of equipment (equalizer, FX processor) is inserted in the signal chain.

Serial Programming

The module can also be programmed using a serial data connection. This is most easily accomplished by using a USB-to-serial chip such as the popular *FT232R* which can be found in many standalone breakout boards and cables, including the *FTDI TTL-232R-3V3* as well as various equivalents from *Adafruit*, *SparkFun*, and others. Data signals should be at 3.3V levels.

The serial connection should be hooked up to the white shrouded 6-pin header labeled **FTDI** (H2) on the module's PCB. This is a *JST-XH* header that works very well with the 0.1" pitch female connectors that are frequently found on these cables. Only pins **1 (GND)**, **4 (RX)**, and **5 (TX)** are used, and the location of pin 1 is marked with a white triangle. Power must be supplied separately via a Eurorack power cable connected to the **POWER** (H3) header.

To prepare the processor for serial programming:

1. Power-on the module.
2. Press and hold both the **BOOT** (S1) and **RESET** (S2) buttons at the same time.
3. Release the **RESET** button.
4. Wait a moment, then release the **BOOT** button.

The processor is now ready to be programmed.

JTAG Programming

The module can also be programmed using a JTAG programmer connected to the black shrouded header labeled **JTAG** (H1) on the module’s PCB. In this scenario, simultaneous ±12V power provided via a Eurorack cable is **not required** since power to the chip is provided via the JTAG programmer itself.

An example of this kind of programmer is the *Olimex ARM-USB-OCD*. Their *ARM-JTAG-20-10* adapter is also necessary in order to accommodate the module’s miniature 0.05″ pitch 10-pin header.

No additional steps are necessary to prepare the module for programming via this method.

If you are interested in serial or JTAG programming, please explore the *Vagrant environment for Mutable Instruments modules hacking* which can be found at **github.com/pichenettes/mutable-dev-environment**

Other programming resources can be found at **forum.mutable-instruments.net/t/4336** (*Mac Tutorial: How to compile and upload the firmware of MIs eurorack modules*) and at **medium.com/music-thing-modular-notes/a08173cec317** (*How to get started writing your own firmware for Mutable Instruments Clouds*), the latter of which was written for the *Clouds* module but is also applicable to *Braids*.

Documentation

Support

For all support inquiries, please send an email to **support@tall-dog.com**

Who We Are

Tall Dog Electronics is located in the Pioneer Valley region of Western Massachusetts. *Tall Dog* has primarily focused on producing a variety of breakout boards for the *Teensy* microprocessor development platform and conducts the majority of its business via the *Tindie* marketplace. *µBraids SE* represents their most ambitious project to date.

Michael Gilbert is a composer, recording artist, and teacher of electronic music, for over 40 years. His music is a creative mix of electronic, jazz, world, and contemporary classical idioms, and is available on 9 albums of original work as *Michael William Gilbert*. The music has featured Adam Holzman, Mark Walker, Peter Kaukonen, David Moss, and Tony Vacca. He has also designed and built electronic music equipment, using it in his own studio and making it available to other musicians.

Daniel Gilbert is a designer and engineer with a background in film, photography, and animation. After graduating from *Hampshire College* he spent the next several years working in the Los Angeles film industry. He now resides in Easthampton, Massachusetts.