Bucket ONE

Software Synthesizer

Version 1.0

http://www.fullbucket.de/music

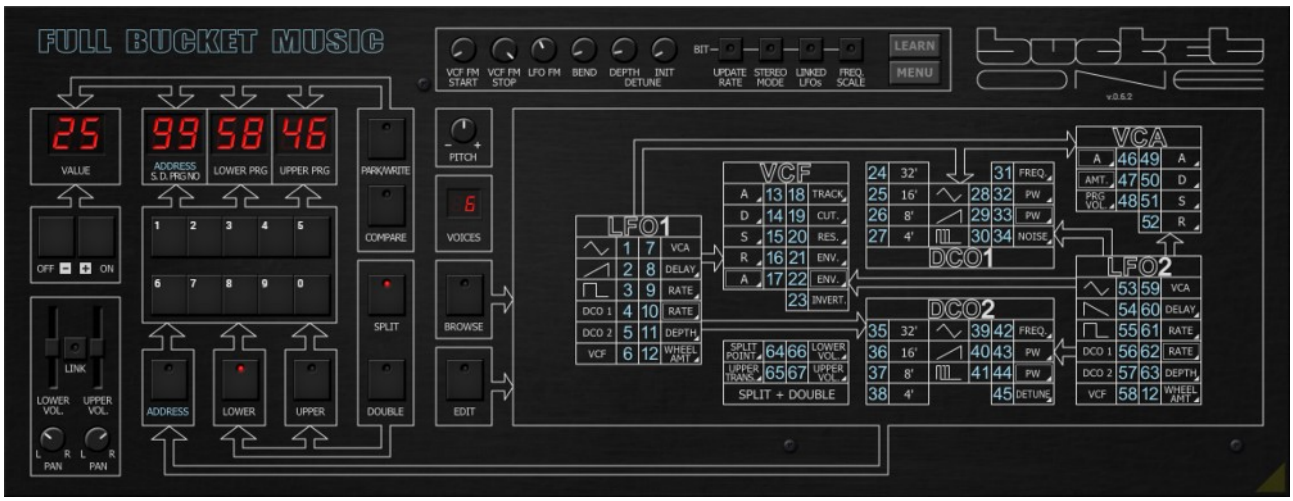


Table of Contents

Introduction

The *Bucket ONE* is a software synthesizer plug-in for Microsoft Windows (VST2/VST3/CLAP) and Apple macOS (VST2/VST3/AU/CLAP) simulating the classic Crumar *BIT 01/99* synthesizers from 1985. It is written in native C++ code for high performance even on "lighter" systems. The main features are:

- Close simulation of the original hardware
- Visual sound editor
- Up to 64 voices polyphony including Split and Double mode
- Two oscillators with three waveforms (triangle, sawtooth, pulse)
- Additional white noise generator
- Classic self-resonating four-pole lowpass filter
- Individual envelopes for VCF and VCA
- Two low frequency oscillators (LFOs)
- MTS-ESP (https://oddsound.com/) dynamic micro-tuning support
- All parameters can be controlled by MIDI controllers
- Plug-in supports Windows and macOS (32 bit and 64 bit)

The *Bucket ONE* is based on the new **iPlug2** framework maintained by **Oli Larkin and the iPlug2 team**. Big thanks, guys!!! Without your work it would not have been possible to create a resizable *Bucket ONE* user interface.

To resize the plug-in you just grab the yellow triangle at the bottom right of the window and drag it. You can save the current window size using the menu entry "Save Window Size" in the *Options Menu*.

If you have trouble with the standard version of the *Bucket ONE*, please grab the (sound-wise identical) "N" version of the plug-in which is based on the original **iPlug** framework.

About This Manual

When you see this 🎧 icon the manual will tell you something about the technical aspects of the *BIT* series and/or the *Bucket ONE* plugin. You may safely skip the respective section then – or not.

YASH – Yet Another Simulated Hardware

Why am I doing this? The answer is simple: Because it is fun. And I sold my good old *BIT ONE* years ago, and I miss it so much (except for the metal side panels close to the keyboard which eventually hurt your pinkies when playing the C1 or C6).

The *BIT ONE* and its successor, the *BIT 01/99* simulated here (the *BIT 01* is the rack version of the *BIT 99*), entered the stage in 1984 and 1985 respectively. Built by the Italian company Crumar which is famous for their string ensembles, synths, and combi keyboards/organs, these units were also sold under the labels "Unique" and "LEM". There was nothing special about the *BITs* except that they were pretty inexpensive and offered keyboard velocity! Wow! [I am not sarcastic here.]

Fun fact: The main developer of the *BIT* synthesizers, Mario Maggi, previously had designed the Elka *Synthex*.

Building A *BIT* Simulation From Bits

Again it's the old dilemma: I want to simulate a unit that I do not own (anymore). Unfortunately, there is not much information available about the *BITs*. Luckily, I found their schematics, their owner's manuals – and images of their firmware ROM chips. Such a ROM contains the program memory of the micro controller unit (MCU, in case of the *BITs* an 8031) that controls a *BIT*.

A bold idea arose in my crazed mind: Why not build a prototype plugin that simulates the MCU plus the hardware of a *BIT* and then model the final *Bucket ONE* plugin according to that prototype?

[They call me mad, but I will show them, ahahaHAHAHAHA!]

For the plugin I chose as base model the *BIT 01* rack expander: The schematics are a little bit simpler, and somehow a plugin is more like a rack expander itself (it does not contain a physical keyboard).

The next step was to translate the byte code of the ROM into a human readable assembler code. There are lots of freeware programs that can do this job – I used DASMx. From this code and the schematics I identified the specific input/output registers controlling the hardware components of the *BIT 01* (DCOs, VCF, amplifiers etc.). A pretty tedious work, especially when you know that assembler code soon gets ugly and contains massives amount of jumps (the technical term is "Spaghetti Code").

The *BIT's* LFOs and envelopes are computed by the MCU in real-time, and the same is true for handling MIDI and program data, as well as buttons and LED displays ("user interface"). Thus, I wrote a piece of code that emulates an 8031 MCU and integrated it into the prototype plugin. After a lot of frustration and joy I finally got a reasonably working *BIT 01* MCU simulator which was the starting point of my *Bucket ONE* plugin.

Why didn't I stop here? Why not releasing the MCU simulator plugin itself but instead going even more steps beyond and building *another* plugin? Here's why:

- The MCU simulator plugin has only 6 voices – like the *BITs*.
- The MCU simulator's CPU load is pretty high.
- Changing sound parameters typically does not affect already playing notes.
- It is buggy as hell.

- More...

I did not dare to extend the original *BIT 01* ROM code to fix these issues, and I really was concerned about the plugin's performance. Finally, it is much more comfortable and effective to write a plugin in "modern" C++!

Words Of Wisdom

So here we are, the *Bucket ONE* awaits you. It has all the quirky things of the ancient machines in it, and feature-wise it is in no way a competitor to modern software instruments. Same is true with respect to usability although I integrated a nice sound editor. And of course my usual warning: It does not sound like the original *BITs*, not even close, it never will, you know, analog fairy dust and all that stuff… ☺

Acknowledgments

- **Oli Larkin** and the **iPlug2** team.
- **kraftraum** (https://soundcloud.com/kraftraum) has designed many of the default patches, and I was able to win him for Beta Testing again – thank you!!!
- **Georg Müller** aka **swissdoc** for providing the factory patches as well as invaluable information about the *BITs*.
- **My family** for bearing me and my crude hobby.

No, I am not affiliated with Crumar (nor KORG) in what relation ever except that I find myself entangled with their instruments. ☺

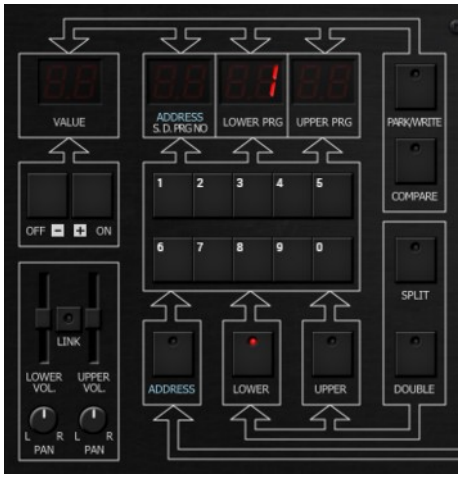
Overview

The *Bucket ONE* like the *BIT 01/99*, features 75 sound programs or presets (called "sound PRGs") and 24 combination programs (called "combi PRGs") which are splitted or stacked combinations of two sound PRGs over the keyboard. Note that the first mode of the *BIT* series, the *BIT ONE*, only has 63 sound PRGs and *no* combi PRGs!

The PRGs themselves don't have names, just numbers (1–75 for sound PRGs and 76–99 for combi PRGs). Those were the 80's.

Furthermore, the *Bucket ONE* has three keyboard modes: SINGLE, DOUBLE, and SPLIT. As you may guess, the DOUBLE and SPLIT modes combine two sound PRGs, the LOWER and the UPPER, and such a combination can be saved as a combi PRG.

Selecting PRGs



Pretty simple: In SINGLE mode click the LOWER button (if not already lit) and then either enter a PRG number using the number button 0 to 9, or click the OFF/– or ON/+ buttons respectively.

When you enter a PRG number greater than 75, you will select a combi PRG (see *below*). This is only possible via the number buttons!

SINGLE Mode

In SINGLE mode you only will play and hear the LOWER sound PRG alone. The *Bucket ONE* starts automatically in SINGLE mode with LOWER sound PRG 1.

SPLIT And DOUBLE Modes

Clicking one of the buttons SPLIT or DOUBLE will activate the respective keyboard mode. Now you can also select an UPPER sound PRG after clicking the UPPER button. To return to SINGLE mode, uncheck the activated SPLIT or DOUBLE button.

There are some parameters controlling the split point and the transpose of the UPPER part in SPLIT mode (see section *SPLIT and DOUBLE Parameters*). You can also adjust the relative volume of the LOWER and UPPER parts (apart from the Volume section, see *Volume Section*)

Combi PRGs

The realm of combi PRGs is entered when you select a PRG number from 76 to 99. Combi PRGs store the values that you can manually set in SPLIT or DOUBLE mode: Split point and UPPER part transpose, LOWER and UPPER part volumes as well as LOWER and UPPER PRG numbers (see section *SPLIT and DOUBLE Parameters*).

Important note:

Combi PRGs do not store sound PRG data!

Assume that you have a combi PRG #99 using the sound PRG #58 for the LOWER part. Now if you change the sound PRG #58, the combi PRG #99 will use that new sound and may itself sound different than expected!

To return to a "normal" sound PRG (in SINGLE mode) you have to hit one of the SPLIT or DOUBLE buttons – whatever is lit.

Comparing And Committing PRGs

Once you have edited a PRG (see section *Editing PRGs*) you can compare it with the original stored PRG – just click the COMPARE button, and click it again to return to your edited sound.

To commit ("write" or "save") a PRG, click the PARK/WRITE button; the current sound data will be "parked" in a temporary space. Now you can select a different PRG (but you don't have to), and when you click the lit PARK/WRITE button again, the sound data of the selected PRG will be overridden with the "parked" sound data you wanted to commit.

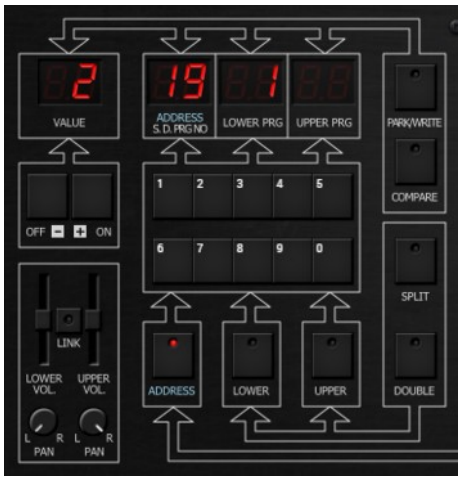
Note that you can cancel the commit action by clicking the COMPARE button.

Editing PRGs

It is 1985. Most of the modern synthesizers have a sophisticated new way of editing sound data: Select a parameter via number buttons and enter the parameter value via up/down buttons. The original *BIT 99* manual states:

> *«Unlike instruments of older design, the BIT 99 does not use a vast array of knobs and buttons, pots and switches to alter and set the various parameters which together shape and form a sound. Instead, modern microprocessor technology takes the heartbreak out of finding a path through the electronic jungle!»*

Let's get this straight: Marketing guys of the 80's wanted to sell us this new pain of editing as the fresh and modern way of the brave new digital world. In fact it was all about saving material cost (the "vast array of buttons, pots, and switches") and drastically reducing the retail price for such a product. Which was a good thing too, or else many folks like me, the poor student Björn, never could have afforded a polyphonic synthesizers like a *BIT*!

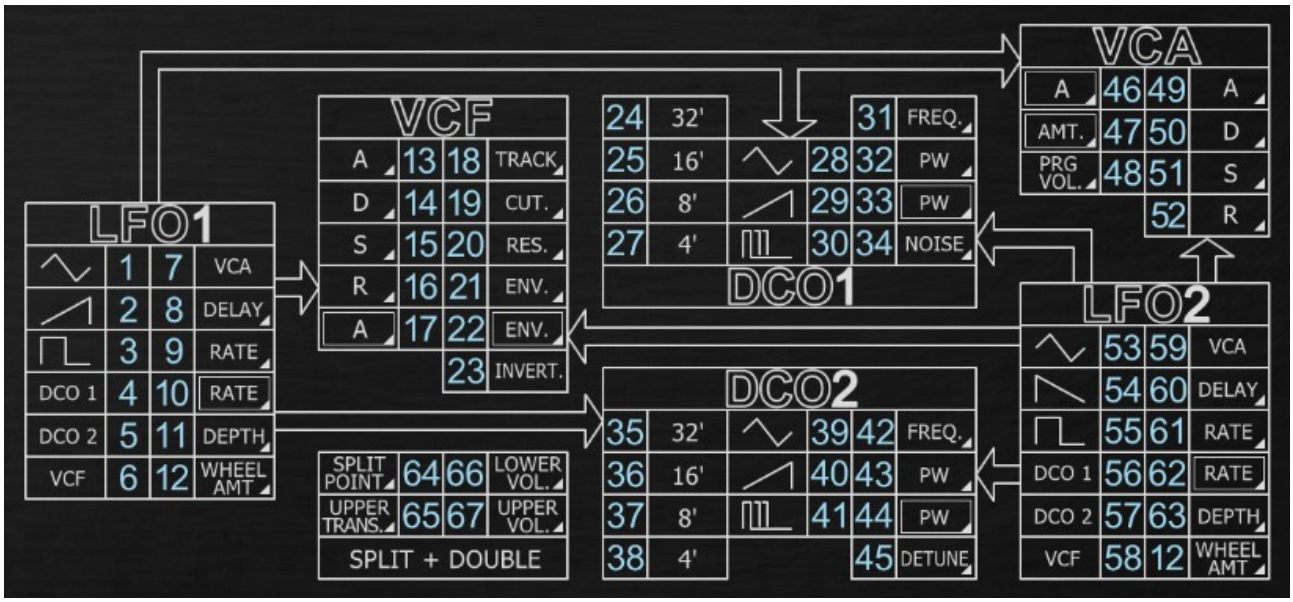


OK, now all you have to do is to change to EDIT mode by clicking the ADDRESS button. Type in the number of a sound parameter using the number buttons and edit the parameter value using the OFF/– and ON/+ buttons (luckily, the values will increase or decrease fast when you "hold" one of

these buttons a bit longer). Given that the *Bucket ONE* has 63 sound parameters and another 4 more for combi PRGs, it really is a PITA.

The *Bucket ONE* (like the *BITs*) has all the parameters and their address numbers displayed in a nice diagram on the main panel of the instrument. When you click on one of those parameters you will immediately be taken to the respective address. Of course this was not possible with the original *BITs*!

The same is true for this handy feature: An edited PRG will be denoted by a red dot in the PRG number display (or in the address display for combi PRGs).



But even more when you click on the EDIT button you get a fancy editor with that vast array of buttons, pots, and switches!

Note that this way you will always edit the LOWER PRG – it is not possible to edit the UPPER PRG directly.

Browsing PRGs

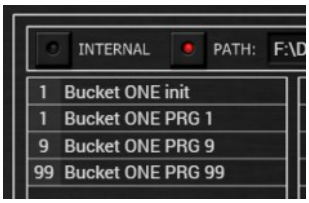
A nice way of selecting PRGs is using the browser feature of the *Bucket ONE*. Clicking the BROWSER button opens the browser panel where you can select one of the factory PRGs or a PRG file stored on your local hard drive.

All the factory PRGs of the *Bucket ONE* are "burned-into" the plug-in itself – there is no way of "losing" them. This is also the reason why they show up with fancy names here (remember that PRGs do not have names but numbers only) – the names are stored in the plug-in as well.

Once you select a PRG in the browser it will switch to the program number shown left to the program's name and set the respective sound data.

When you click the PATH button you will see (or not) the PRGs that are stored at the given path of your local drive (you can change the path by clicking the text box showing the path itself). These are MIDI System Exclusive (.syx) files in the original *BIT 01/99* format.

The *Bucket ONE* tries to parse them and only shows "valid" sound files along with their respective program numbers and the file names (without the ".syx" extension). Thus, you can come up with nice PRG names by choosing decent names for your sound files. A counterexample is shown in the image to the right. ☺



Volume Section

It looks straight forward, but of course there is a twist: You can set the overall volume and stereo panorama for both the LOWER and UPPER part individually (when the LINK button is lit, the respective sliders and knobs are linked and are conveniently set together). This makes perfect sense in SPLIT and DOUBLE mode, but what happens in SINGLE mode? Naively one might think that only the LOWER controls would have any effect on the sound, but that is not true: In fact the LOWER and UPPER slider/knobs control the volume and panorama of the respective LOWER and UPPER voices (see next section *LOWER And UPPER Voices*)

LOWER And UPPER Voices

Technically speaking, the *Bucket ONE* and the *BITs* have two sets of voices: One set for the LOWER and one for the UPPER part. For the *BIT 01/99*, the number of voices per part is 3 which makes in sum 6. Thus, in SINGLE mode all the voices of both parts are combined to make up a 6-voice polyphonic instrument.

Depending on the number of voices selected (6, 16, 32, or 64), the *Bucket ONE* can have more voices for the LOWER and UPPER parts (3, 8, 16, or 32). But anyway, since the voices are organized in two parts, the volume and panorama or their part's sum can be controlled individually, too.

Now in SINGLE mode the question is how notes are scheduled to voices: Will a note be played by a LOWER or an UPPER voice? This can have some funny side effects since volume and panorama of the parts might be set differently!

By default, the *Bucket ONE* schedules each new note to a different part than the note before – the voice scheduling mode is *alternating*: The first note goes to LOWER, the second note to UPPER, the third again to LOWER and so on. This pattern gives some nice stereo effect when you select a different panorama setting for the LOWER and UPPER parts.

The *BITs* behave different since they schedule the *first three* notes to the LOWER and the *last three* to the UPPER part. This might sound a bit odd when you use heavy stereo panning. Anyway you can set the *Bucket ONE* to that mode, too – see next section.

Control Section

In the control section you will find some global parameters that may or may not affect the sound generation of the *Bucket ONE*.

- **VCF FM START** and **STOP**:

These two parameters control the *minimum* and *maximum* value of the VCF's cutoff frequency when modulated by any modulation source (envelope, key track, velocity, LFOs). Within the original *BITs* those values depend (among others) on the setting of many trimming potentiometers and may change from model to model. However, the default values of the parameters are taken as "best guesses", and you might never need to change them.

- **LFO FM** and **BEND**

These two parameters set the maximum amount of frequency modulation of the DCOs by the LFOs and/or Pitch Bend information (in half note steps). The reasoning is the same as above.

- **DETUNE DEPTH** and **INIT**

Same thing for the DEPTH of the DETUNE parameter. And the DETUNE INIT determines the initial amount of detuning of DCO 1 and 2 when the DETUNE

parameter is set to 0. There always will be some slight detuning of the two DCOs in the *BIT* hardware, even though they are *digitally controlled*. This is because the two master oscillators of the 6 + 6 DCOs never are in perfect tune – they are made of uncoupled analog circuitry.

[What? The *BITs* use master oscillators? Are the *BITs* just better organs using frequency divider circuitry? Yes, almost: The frequency dividers are standard 8253 chips featuring programmable counters/dividers. But to say that the *BITs* are "just better organs" is highly unfair!]

- **UPDATE RATE**

Real-time processing inside synthesizers like the *BIT* means that the normal work of the internal processor is interrupted in regular intervals to manage incoming MIDI data, update envelopes and LFOs etc. The MCU 8031 inside the *BITs* isn't the fastest in the world, and thus the internal update rate is about 325 Hz. You can set the *Bucket ONE*'s update rate to this (lower) value by checking the UPDATE RATE button.

- **STEREO MODE**

As described in section *LOWER And UPPER Voices* the *Bucket ONE* schedules incoming notes to alternating voice parts (LOWER or UPPER). When you check this button, the part scheduling of the original *BITs* will be applied.

- **LINKED LFOs**

There is another secret that no one has told you before: The *BITs* indeed do only have two LFOs – for all 6 voices **and** for the two parts (LOWER and UPPER) together. Now what happens when you register two different sound PRGs in SPLIT or DOUBLE mode? Well, in this case the LFO settings of the LOWER PRG are used for the UPPER PRG, too. Thus, your UPPER sound might sound totally different than when you select it in SINGLE mode!

This was always confusing me when I played my good old *BIT ONE* back in the days. And that's why the *Bucket ONE* by default uses two sets of LFOs per part (four LFOs in total), so even the UPPER sound will get its own and correct LFO settings. But if you want to change back to the original behavior: Check the LINKED LFOs button.

- **FREQ SCALE**

The use of frequency divider technology (see above) has its drawbacks. One is that you practically cannot cover the whole frequency range of about 10 MIDI octaves, and the other is that you cannot set the *exact* frequency values for each note. Luckily the *BITs* were laid out for 5-octave keyboards… check the FREQ SCALE button to hear how frequency scaling was originally done.

- **LEARN and MENU**

Those buttons will be described below.

Voice Architecture

It is pretty simple: Two DCOs going into a single VCF and a single VCA with individual envelope generators plus two global LFOs.

DCOs

I personally think that the term "DCO" – Digitally Controlled Oscillator – is a misnomer. In fact the signal generated by an oscillator of the *BIT* series is derived directly from a 5 bit digital-analog converter (DAC) driven by a modulo counter. The result is a "staircase sawtooth", and the two other available waveforms (triangle and pulse) are created via additional analog circuitry. Due to that treatment the triangle signal is very "rough" and sounds a bit dirty.

The nice thing is that all three waveforms of an DCO can be combined (yet not continuously mixed). The pulse width can be controlled manual and via velocity information – but not via the LFOs.

Besides the frequency range (32', 16', 8', 4') you can also set an additional interval (FREQ.) per oscillator. DCO 1 also allows you to add some NOISE while DCO 2 features the DETUNE parameter.

VCF

The VCF (Voltage Controlled Filter) of a *BIT*'s voice, a four-pole lowpass, is realized using a CEM 3328 chip which is a low-cost variant of the CEM 3320 that was built into classic instruments like the *OB-Xa*, *Synthex*, and many many more. The *Bucket ONE* humbly tries to recreate it and fails. ☺

Parameters are almost standard: Cutoff frequency, resonance, keyboard tracking, envelope intensity, and velocity-controlled envelope intensity (nice!). You can invert the whole frequency modulation stuff, and you have a dedicated ADSR envelope where the Attack time can be modulated by velocity, too!

VCA

For the VCA (Voltage Controlled Amplifier) you can set the program level and velocity intensity. It has its own ADSR which is identical to the VCF's envelope (including velocity-controlled Attack time).

LFO 1 and 2

There is one global LFO 1 (Low Frequency Oscillator) and one LFO 2 for the whole instrument (well, the *Bucket ONE* by default has one pair of LFOs per part, see section *Control Section*). The only difference is that LFO 1 offers a *rising* Sawtooth while LFO 2 instead has a *falling* one. Alternatively you can select a Triangle or a Square wave, or switch the LFO off.

The modulation targets are DCO 1 and DCO 2 frequency, VCF cutoff frequency and VCA level, but unfortunately the modulation depth cannot be set individually per destination. The same is true for the intensity of the MIDI Modulation Wheel controlling the LFO modulation, and this intensity has to be set for both LFOs together.

However, you have the option to set a delay time until the full modulation amount builds up. Since the LFOs are *global* and not per voice, the delay function is re-triggered with each note key pressed.

Another very impressive feature is that the rate of an LFO can be controlled by velocity.

SPLIT and DOUBLE Parameters

In SPLIT and DOUBLE mode you will find four more parameters: LOWER and UPPER volume level, SPLIT POINT and UPPER TRANS(POSE).

Some words about the latter two parameters which only have significance in SPLIT mode. Their values range from 1 to 61 with respect to the 61 keys on a five-octave keyboard. "1" represents C1 where C1 = MIDI note number 36 (the lowest note on a *BIT 99* keyboard), and "61" represents C6 = MIDI note number 96 (the highest note on a *BIT 99* keyboard). Thus, the SPLIT POINT parameter defines the first key on the keyboard where the UPPER part "starts to play".

Now for the UPPER TRANS parameter things get a little bit complicated: Its parameter value actually denotes which note will be played for the first note of the UPPER part i.e. the note at the split point. Say that SPLIT POINT is set to "25" (C3 = MIDI note number 60) and UPPER TRANS to "25", then the note actually played by the UPPER part at the split point is C3 – the UPPER part is not transposed at all. When you set UPPER TRANS to "1", then the note played at the split point is C1 – the UPPER part is transpose two octaves downwards. On the other hand, UPPER TRANS = "61" leads to a 3-octave upwards transposition, but when you set SPLIT POINT to "1" this changes to a full 5-octave transposition!

While this is technically logical, it is also musically very counter-intuitive, and UPPER TRANS(POSE) is something like a misnomer. Remember: The value set at UPPER TRANS corresponds to the keyboard note (C1 to C6) played for the first note of the UPPER part at the SPLIT POINT. Ouch.

Plugin Handling

Loading And Saving PRGs

As stated earlier (see section *Browsing PRGs*) the *Bucket ONE* reads and saves PRG data as MIDI System Exclusive files with file extension ".syx" in the original *BIT 01/99* format. You can read/save PRGs from menus at various places, for example from the *Options Menu* or by right-clicking the LOWER, UPPER and ADDRESS LED display. Depending on the context, the context menus (sic!) may not always show all entries or might not open at all – the *Bucket ONE* figures out what can be done in the current situation and tries not to flood you with inaccessible menu entries.

Options Menu

When clicking the *Menu* button in the Control section (see *Control Section*), a context menu opens with the following options:

Init PRG	Initialize the current LOWER PRG
Load Lower PRG	Load a System Exclusive file containing sound data to the *Bucket ONE's* current LOWER PRG
Load Upper PRG	Only available in SPLIT or DOUBLE mode: Load a System Exclusive file containing sound data to the *Bucket ONE's* current UPPER PRG
Load Combi PRG	Only available when a combi PRG is selected: Load a System Exclusive file containing combination data to the *Bucket ONE's* current combi PRG
Load SysEx File	Load a System Exclusive file into the *Bucket ONE* and see what's happening
Restore Factory PRGs	Restore the original factory PRGs
Save Lower PRG	Save the *Bucket ONE's* current LOWER PRG to a System Exclusive file
Save Upper PRG	Only available in SPLIT or DOUBLE mode: Save the *Bucket ONE's* current UPPER PRG to a System Exclusive file
Save Lower PRG	Only available when a combi PRG is selected: Save the *Bucket ONE's* current combi PRG to a System Exclusive file
Save All PRGs As Single File	Saves all sound and combi PRGs consecutively in one big System Exclusive file. You can reload that file using the "Load SysEx File" function above.
Default Path for PRG Files...	Sets the default path for reading and saving PRG files
Default Path for SysEx Files...	Sets the default path for reading and saving System Exclusive files
MIDI Thru	Set globally if MIDI data sent to the *Bucket ONE* should

	be sent through to its MIDI output
Ignore Program Change	Set globally if MIDI Program Change data sent to the *Bucket ONE* should be ignored
Reload Configuration	Reload the *Bucket ONE's* configuration file (see section *The bucketone.ini Configuration File*)
Save Configuration	Save the *Bucket ONE's* configuration file (see section *The bucketone.ini Configuration File*)
Window Size...	Change the window size of the *Bucket ONE*
Save Window Size	Stores the current window size to the configuration file so that it will be restored next time you load the *Bucket ONE*
Check Online for Update	When connected to the Internet, this function will check if a newer version of the *Bucket ONE* is available at fullbucket.de
Visit fullbucket.de	Open fullbucket.de in your standard browser

The *bucketone.ini* Configuration File

The *Bucket ONE* is able to read some settings from a configuration file (`bucketone.ini`). The exact location of this file depends on your operating system and will be displayed when you click on "Reload" or "Save Configuration".

MIDI Control Change Messages

All parameters of the *Bucket ONE* can be controlled by MIDI controllers, or more precise: Each MIDI controller (except *Modulation Wheel* and *Sustain Pedal*) can control one of *Bucket ONE*'s parameters. The mapping is defined in the `bucketone.ini` for example like this:

```
[MIDI Control]
CC7  = 4  # Volume
CC70 = 32 # Filter 1 Cutoff
CC71 = 33 # Filter 1 Resonance
...
```

The syntax is straight forward:

```
CC<controller number> = <parameter ID>
```

Given the above example, controller 7 directly controls the overall *Volume* parameter, controller 74 the *Filter 1 Cutoff* etc. As you can see, comments are introduced by the Pound sign (#); they are here just for description purposes and completely optional. Note that the *controller number* can run from 0 to 110, with the exception of 1 (*Modulation Wheel*) and 64 (*Sustain Pedal*); the latter two are simply ignored.

MIDI Learn

The easiest way to assign MIDI controllers to *Bucket ONE* parameters is to use the *MIDI Learn* function. To activate MIDI Learn, click on the LEARN button and wiggle

both the MIDI controller and the *Bucket ONE's* parameter that you want to link. If you want to unlearn the assignment, right-click the LEARN button (the label now reads "UNLRN") and activate it. Now wiggle the MIDI controller or the parameter that you want to unlearn.

Plug-In Parameters

The Plug-in parameters represent all the parameters that are available on the original *BITs* and printed on their main panel. However, they are organized a little bit more efficient (see for example *LFO 1+2 Waveform* and *DCO 1+2 Octave*), and their IDs do not match the addresses of the panel parameters.

LFO 1

parameter	id	description
Waveform	0	Waveform (0 = Triangle, 1 = rising Sawtooth, 2 = Square)
DCO 1	1	DCO 1 frequency modulation (0 = off, 1 = on)
DCO 2	2	DCO 2 frequency modulation (0 = off, 1 = on)
VCF	3	VCF cutoff frequency modulation (0 = off, 1 = on)
VCA	4	VCA level modulation (0 = off, 1 = on)
Delay	5	Delay time (0 – 63)
Rate	6	LFO Rate (0 – 63)
Rate Dynamic	7	Intensity of Rate modulation by velocity (0 – 63)
Depth	8	Modulation amount (0 – 63)
Wheel Amount	9	Intensity of Depth modulation by Mod. Wheel (0 – 63)

VCF

parameter	id	description
Attack	10	Attack time of envelope (0 – 63)
Decay	11	Decay time of envelope (0 – 63)
Sustain	12	Sustain level of envelope (0 – 63)
Release	13	Release time of envelope (0 – 63)
Attack Dynamic	14	Intensity of Attack time modulation by velocity (0 – 63)
Track	15	Keyboard tracking (0 – 63)
Cutoff	16	Cutoff frequency (0 – 63)
Resonance	17	Resonance (0 – 63)
Envelope	18	Envelope amount
Envelope Dynamic	19	Intensity of Envelope modulation by velocity (0 – 63)
Invert Envelope	20	Inversion of cutoff frequency modulation (0 = off, 1 = on)

DCO 1

parameter	id	description
Octave	21	Octave (0 = 32’, 1 = 16’, 2 = 8’, 3 = 4’)
Triangle	22	Triangle wave (0 = off, 1 = on)
Sawtooth	23	Sawtooth wave (0 = off, 1 = on)
Pulse	24	Pulse wave (0 = off, 1 = on)
Frequency	25	Frequency interval (0 – 11 notes)
Pulse Width	26	Pulse width (0 – 30)
Pulse Width Dyn.	27	Intensity of Pulse Width modulation by velocity (0 – 63)
Noise	28	Noise level (0 – 63)

DCO 2

parameter	id	description
Octave	29	Octave (0 = 32’, 1 = 16’, 2 = 8’, 3 = 4’)
Triangle	30	Triangle wave (0 = off, 1 = on)
Sawtooth	31	Sawtooth wave (0 = off, 1 = on)
Pulse	32	Pulse wave (0 = off, 1 = on)
Frequency	33	Frequency interval (0 – 11 notes)
Pulse Width	34	Pulse width (0 – 30)
Pulse Width Dyn.	35	Intensity of Pulse Width modulation by velocity (0 – 63)
Detune	36	Detune (0 – 63)

VCA

parameter	id	description
Attack Dynamic	37	Intensity of Attack time modulation by velocity (0 – 63)
Amount Dynamic	38	Intensity of VCA level modulation by velocity (0 – 63)
Program Volume	39	Programmable volume of PRG (0 – 63)
Attack	40	Attack time of envelope (0 – 63)
Decay	41	Decay time of envelope (0 – 63)
Sustain	42	Sustain level of envelope (0 – 63)
Release	43	Release time of envelope (0 – 63)

LFO 2

parameter	id	description
Waveform	44	Waveform (0 = Triangle, 1 = falling Sawtooth, 2 = Square)
DCO 1	45	DCO 1 frequency modulation (0 = off, 1 = on)
DCO 2	46	DCO 2 frequency modulation (0 = off, 1 = on)
VCF	47	VCF cutoff frequency modulation (0 = off, 1 = on)
VCA	48	VCA level modulation (0 = off, 1 = on)
Delay	49	Delay time (0 – 63)
Rate	50	LFO Rate (0 – 63)
Rate Dynamic	51	Intensity of Rate modulation by velocity (0 – 63)
Depth	52	Modulation amount (0 – 63)

SPLIT + DOUBLE

parameter	id	description
Split Point	53	Split point (1 – 61; SPLIT mode only)
Upper Transpose	54	Transpose of UPPER PRG (1 – 61; SPLIT mode only)
Lower Volume	55	Programmable LOWER volume (0 – 63)
Upper Volume	56	Programmable UPPER volume (0 – 63)

Extension Parameters

These parameters are global parameters that are used to manage the plug-in.

parameter	id	description
Master Tune	57	Master tune (± 1 note)
Keyboard Mode	58	Keyboard mode (*Single*, *Split*, *Double*)
Lower Volume	59	Lower part volume (fader)
Upper Volume	60	Upper part volume (fader)
Lower Pan	61	Lower part panorama
Upper Pan	62	Upper part panorama
Program Number	63	Current PRG number (equals Lower PRG number in SINGLE mode and combi PRG number for a combi PRG)
Lower Program	64	Current Lower PRG number
Upper Program	65	Current Upper PRG number
Voices	66	Number of polyphonic voices (5, 16, 32, 64)
Update Rate	67	Update rate (*High*, *BIT*)
Stereo Mode	68	Stereo Mode (*Alternating*, *BIT*)
Linked LFOs	69	Linked Lower + Upper LFOs (*Off*, *On*)
Frequency Scale	70	Frequency scale (*Equal*, *BIT*)
VCF FM Start	71	Minimum cutoff freq. of VCF modulation (8.2 – 19912 Hz)
VCF FM Stop	72	Maximum cutoff freq. of VCF modulation (8.2 – 19912 Hz)
Pitch Bend Range	73	Pitch Bend range (0 – 24 notes)
LFO FM Range	74	LFO to DCO frequency modulation range (0 – 12 notes)
Detune Depth	75	Maximum Detune of DCO 2 (0 – 2 notes)
Detune Init	76	Initial Detuneof DCO2 (0 – 0.1 notes)

Frequently Asked Questions

How do I install the Bucket ONE *(Windows VST2 32 bit version)?*

Just copy the files `bucketone.dll` and `bucketone.ini` from the ZIP archive you have downloaded to your system's or favorite DAW's VST2 plug-in folder. Your DAW should automatically register the *Bucket ONE* VST2 plug-in the next time you start it.

How do I install the Bucket ONE *(Windows VST2 64 bit version)?*

Just copy the file `bucketone64.dll` and `bucketone.ini` from the ZIP archive you have downloaded to your system's or favorite DAW's VST2 plug-in folder. Your DAW should automatically register the *Bucket ONE* VST2 plug-in the next time you start it.

Note: You may have to remove any existing (32 bit) `bucketone.dll` from your VST2 plug-in folder or else your DAW may screw the versions up…

How do I install the Bucket ONE *(Windows VST3 64 bit version)?*

Just copy the files `bucketone.vst3` from the ZIP archive you have downloaded to your system's or favorite DAW's VST3 plug-in folder. Your DAW should automatically register the *Bucket ONE* VST3 plug-in the next time you start it.

How do I install the Bucket ONE *(Mac VST2/VST3/AU 64 bit)?*

Locate the downloaded PKG package file `bucketone_1_0_0_mac.pkg` in Finder (!) and do a right- or control-click on it. In the context menu, click on "Open". You will be asked if you really want to install the package because it comes from an "unidentified developer" (me ☺). Click "OK" and follow the installation instructions.

What is the plug-in ID of the Bucket ONE*?*

The ID is **`fb-1`** .

Come on, Bucketeer, yet another boring simulation?

Yup, nostalgia. If you get bored, don't download it. The trash bin is your friend, too.

Where are all the PRGs stored?

All the 99 PRGs are stored along with the state of the plug-in. If you have two instance of the *Bucket ONE* in a DAW project then each of those instance comes with its own set of 99 PRGs – the two sets will not interfere with each other.

Huh? Suddenly my combi PRG sounds different than before!?

See section *Combi PRGs*: Combi PRGs do not store sound PRG data but only the parameters of the SPLIT and DOUBLE modes.

The browser list is empty when I click the "Path" button!

In this case there are no *Bucket ONE*-compatible System Exclusive at the respective path.