

FLASH



THE MINI·POLY·MIDI·MICRO·SYNTH

H π INSTRUMENTS

Aaron Andrew Hunt

Change Log	3
v7.5 - August 25, 2023	3
Precautions	4
Location	4
Power Requirements	4
Handling	4
Electrical Grounding	5
Cleaning	5
FCC REGULATION.....	5
DISPOSAL.....	5
Introduction	6
History	6
Manufacture	6
FLASH-Programmer.....	7
Software (Universal Tuning Editor).....	7
Current Firmware – v7.....	7
Using This Documentation	7
Specification	8
MIDI	9
MIDI Channels.....	9
Continuous Controllers (CC).....	9
Program Changes (Patches).....	10
Tuning	11
Pitch Bend	11
Tuning Tables	11
Synthesis	14
Algorithms	14
Waveforms.....	15
Envelopes	16
Low Frequency Oscillator (LFO) and Noise Source.....	17
Detune	17
Portamento and Arpeggiator	17
Output Gain (advanced)	17
Algorithm 1	17
Algorithm 2.....	19
Algorithm 3	20

<i>Band-limited Polysynth</i>	20
<i>Polysynth Controls</i>	21
<i>Band-limited Monosynth (pseudo-moog)</i>	21
<i>Monosynth Controls</i>	21
Software – UTE	23
<i>Select FLASH under Devices</i>	23
<i>Open the FLASH Window</i>	23
<i>Bootloader vs. MIDI Function</i>	24
<i>Connecting the Programmer</i>	24
<i>Firmware Version & Updating</i>	24
<i>Open Source Firmware</i>	25
<i>MIDI Input, CC Map, & MIDI Learn</i>	25
<i>Patch Basis</i>	26
<i>Editing a Patch & storing that as a new Patch</i>	26
<i>Startup Patch</i>	27
<i>MIDI Receive Channel(s)</i>	27
<i>Tuning Tables</i>	27
<i>Startup Tunings</i>	28
Troubleshooting	29
Appendix	31
<i>CC26 and the “Hammondish” Waveform Drawbars</i>	31
Previous Versions Change Log	32
<i>v7 - August 10, 2022</i>	32
<i>v6 - November 27, 2020</i>	32
<i>v5 - October 30, 2020</i>	32
<i>v1.4 - March 23, 2020</i>	32
<i>v1.3 - January 20, 2020</i>	32
<i>v1.2 - January 15, 2020</i>	33
<i>v1.1 - January 13, 2020</i>	33
<i>v0.9 - December 1, 2019</i>	33
Credits	34

Change Log

Here are lists of changes for this version of the documentation. Changes for previous versions can be found in the section *Previous Versions Change Log*.

Please report typos or problems with this text via email to hpiinstruments@zentral.zone

v7.5 - August 25, 2023

- Added information on firmware v7 features CC 119 for patch selection and startup patch selection to *MIDI*
- Added section on firmware v7 *Startup Patch* to *Software – UTE*
- Edited section on *Startup Tunings* in *Software – UTE*
- Moved change logs for previous versions of this documentation to a new section called *Previous Versions Change Log*.

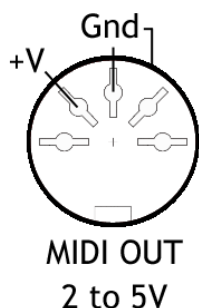
Precautions

Location

Using the synth or programmer in the following locations can result in damage.

- In prolonged direct sunlight
- In extreme temperature or humidity
- In excessively dusty or dirty locations
- In excessive vibration
- In close proximity to magnetic fields

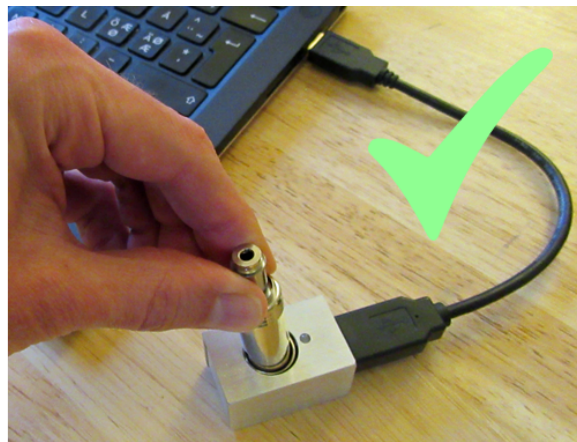
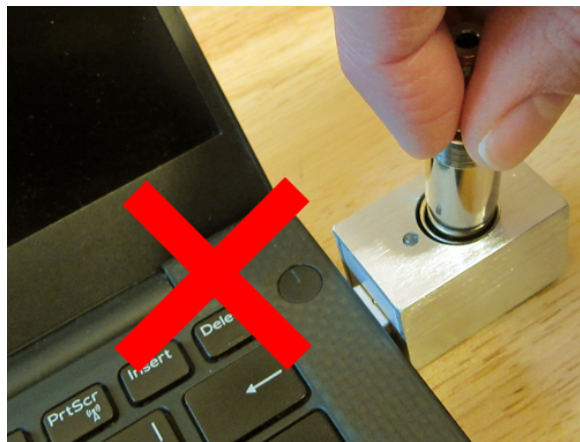
Power Requirements



WARNING – *Not all MIDI OUT ports are made according to the MIDI standard. A non-standard socket could damage the synth, the socket, or both!* FLASH draws its power directly from a standard MIDI OUT socket where pin 2 (centre) is ground, and pin 4 (left of centre) is positive voltage between 2 and 5 Volts. If you have experience with electronics testing, you can check the jack of your instrument using a multi-meter. Plug in a MIDI cable and measure the pins as shown for voltage and amperage output. The pins of the attached cable will match the pins of the jack as shown. *Be careful not to short the pins when measuring!*

Handling

The FLASH synth housing is made of zinc alloy. The programmer housing is made of aluminium (sturdy but softer than zinc). Do not physically abuse the units. If the programmer is plugged in directly into a USB socket, the synth should *NOT* be plugged or unplugged from the DIN socket, because perpendicular force could break the USB connector or port on the computer!



If you want to plug and unplug the synth while the programmer is plugged into USB, always use the included USB extension cable!

Electrical Grounding

Do not plug the audio output of the synth into the same device providing MIDI output to the synth, as this will cause a ground loop stopping the synth from functioning properly. Some MIDI controllers which read positive voltage output as described above still may not provide proper grounding for the synth to function. In some cases using a *Ground Loop Isolator* on the audio output from the synth may fix the problem.

Cleaning

Clean the housing of the units with a dry cloth. For your safety, please do not use liquid cleaners or flammable polishes.

FCC REGULATION



This device complies with Part 15 of FCC Rules (USA). Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

DISPOSAL



Disposal of this product, manual, or package must be done in an approved manner (EU). Do not discard this product, manual, or package along with ordinary household waste. Disposing in the correct manner will prevent harm to human health and potential damage to the environment. The correct method of disposal depends on the applicable laws and regulations in your locality.

Introduction

FLASH is a unique portable and durable MIDI-powered polyphonic microtonal FM synthesizer in a DIN plug, made for on-the-go musicians with a love of gadgets and a flair for style.



History

The FLASH synthesizer is based on previous work done by Tim Alex Jacobs (a.k.a. mitxela), inventor of “The World’s Smallest MIDI Synthesizer”. The project began in October 2017, initiated and funded by Aaron Andrew Hunt (AAH, H-Pi Instruments). Initial specification was for a 16-voice polyphonic microtonal synthesizer supporting independent per-channel pitch bending as well as storage of at least 16 tuning tables. During two years of development, this specification was realised and expanded to take ultimate shape as a heartfelt tribute to the almost-half-century-old MIDI DIN plug, with a nod to the modern standard USB in the form of a concurrently designed UART-USB serial programming bridge device (the FLASH-programmer), produced by Jordan Dimitrov Petkov (JDP † 2020). Pre-orders for the synth and programmer began through the H-Pi website in September 2019, with an initial run of 150 units selling out in a matter of weeks. Since then, production has continued in small quantities released about every other month.

Manufacture

FLASH is not a mass produced product. Each unit is made by hand in Germany and Bulgaria by AAH and JDP. Every synth is individually “brought to life” with its accompanying UART programmer assembled, tested and shipped from Germany by AAH.

FLASH-Programmer

Along with FLASH, you get a device used for reading and writing patches, tunings, and firmware. This device may be referred to by several names. Mac and Windows computers recognise it as “FLASH-programmer”. Technically speaking, it is a “USB-to-UART Serial Bridge”, or a “UART module” having a DIN5 (MIDI) socket to host the synth. In this documentation we may also simply call it “the programmer”.



When the programmer is plugged in, its LED should flash twice, then remain unlit. The LED then will flash only when data is transferred. If the device is not recognised by your system, or does not work properly, it is a driver problem. On Windows the programmer driver will install automatically when you plug in the device for the first time. On most Mac systems a driver is not needed, but if there is a problem, a driver may need to be installed manually; see <https://hpi.zentral.zone/flash>

It is important to note that the programmer is *not* a MIDI interface, although it can be used to send MIDI data. On MacOS, a software bridge can be set up to use the programmer as a MIDI-OUT interface to the synth, to control the synth in real time from any MIDI-capable software. This can also be done on Windows using a virtual driver such as LoopB1, but latency is high, so the programmer is not suitable for real-time MIDI use on Windows.

Software (Universal Tuning Editor)

Purchase of FLASH also includes a free software license for Universal Tuning Editor (UTE), which runs on MacOS and Windows. UTE works with the FLASH-Programmer to write and read firmware, tunings, and patches to and from the synth, and to send MIDI data to the synth. Details are in the Chapter **Software – UTE**. Your license for UTE was sent to you in an email at the point of sale. If you did not receive this email, log into your account at <https://hpi.zentral.zone/login> (or create an account and then log in) and your license code will appear in your account under *my licenses*. If it doesn't appear, contact us by email.

Current Firmware – v7

This documentation has been written about FLASH firmware v7. The firmware may be updated at any time, and posted at <https://hpi.zentral.zone/flash> Step-by-step instructions for updating firmware are given in **Software – UTE**.

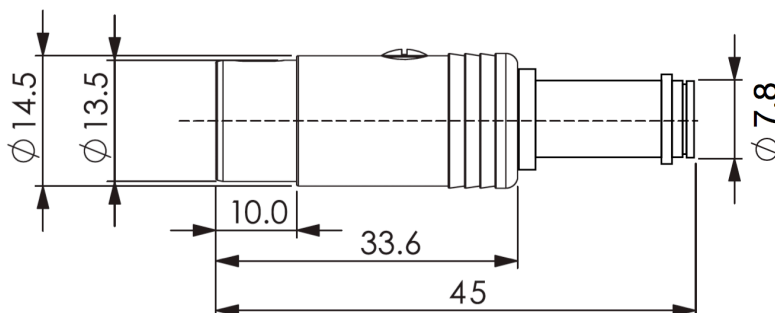
Using This Documentation

Please read through this text completely. It is written to be read from beginning to end, with cross referencing for normal use.

Please report typos or problems with this text via email to hpiinstruments@zentral.zone

Specification

The following technical data is supplied for the synthesizer and programmer:



- **Physical Dimensions** – FLASH Synth: 45mm x 14.5mm, FLASH-programmer: 53mm x 26mm x 25mm
- **Weight** – FLASH Synth: 24g, FLASH-programmer: 35g
- **Power Consumption** (FLASH Synth, maximum with output load) – around 27mW

All below-listed dB output values depend on the MIDI keyboard powering FLASH, and the output load connected to it (headphones/speakers).

- **dBm (power output)** – around 14dBm.
- **dBV or dBu (voltage output level)** – typically between 2.0V and 2.5V, giving a peak-to-peak value of around 0.14V for polyphony of 16 voices without compression, with a real RMS value depending on patch waveform.
- **dB SPL (sound pressure level output)** – depends on the speakers or headphones (impedance and the physical design), but since the synth is being powered from a MIDI connector, the answer is "fairly quiet".

In our beta testing, the headphones output volume level was considered reasonable given the MIDI-powered nature of the synth, but you will probably want to amplify the output the same way you would amplify output from any other synthesizer, using a standard instrument amplifier or powered speakers such as those used with desktop computers. Many portable headphone amplifiers are also available.

For extended use with headphones we get excellent results using the FiiO A1 portable headphone amplifier – <https://fiio.com/a1>

MIDI

MIDI Channels

MIDI data gets sent out from your controller (usually a keyboard), over a MIDI Channel or over Channels. When “MIDI Channel” is mentioned in this text, it is referring to the Channel or Channels on which your MIDI controller is sending MIDI data. FLASH is not a multi-timbral synthesizer, but by default it responds to incoming MIDI on all 16 MIDI channels, respecting Pitch Bend on each channel separately so that each note of the polyphony output from FLASH can be individually tuned. FLASH is unique in this respect, allowing it to be used with unconventional microtonal keyboards like the Tonal Plexus without requiring the use of internal Tuning Tables in the synth. A standard MIDI keyboard will normally send MIDI over one channel, but you may also be able to assign splits or zones for sending MIDI over multiple channels.*

** Starting with firmware v4, FLASH can be programmed (using UTE) to receive on only one MIDI channel if needed – see **Software – UTE > MIDI Receive Channel(s)***

Continuous Controllers (CC)

MIDI Continuous Controller (CC) messages are the language of knobs and sliders, and sometimes switches. A CC message consists of a controller number, and a value ranging from 0 to 127. FLASH responds to the following CC messages received.*

CC	Description	CC	Description
1	Modulation (LFO Depth)	28	Pulse Wave Modulation (Alg3 only)
76	Modulation (LFO Frequency)	24	Algorithm Select
5	Portamento (MONO only)	25	Waveform Select (Alg3: OSC Config)
64	Sustain (0 = off, 64 = on)	26	Wave Parameter Control
20	FM Ratio (coarse, Alg3: Env. 1)	3	Per-channel Tuning
21	FM Ratio (fine, Alg3: Filter Res.)	9	All-channels Tuning
75	Attack Rate (Env. 1)	27	Arpeggiator Speed (MONO only)
22	Effect Depth (Env. 1)	15	Oscillator Detune
23	Effect Decay (Env. 1)	88	Output Gain
72	Release Rate (Env. 2)	100, 101	RPN Control Number
73	Attack Rate (Env. 2)	6, 38	RPN Data Entry
		119	Patch Change**

Program Changes (Patches)

A program change or patch is a collection of all the parameters controlling the sound of the synth. The synth stores 128 patches in non-volatile memory, which can be read, edited, and stored using the included FLASH-Programmer and UTE (see **Software – UTE**). Patches are selected by sending a MIDI Program Change message to the synth on any activated channel. An easy way to get started is to select a patch, then start sending some MIDI CC messages to adjust the sound. Firmwares v2 through v5 include default patches 0 - 33 as listed in the table below. Patches 34 through 38 are introduced with firmware v6 to demonstrate Algorithm 3.

Patch	Patch	Patch
0 Electric Piano 1	13 Sci-Fi 3	26 Bass 1
1 Patch 1	14 Smooth Lead	27 Harmonic Progression
2 Cello	15 Lead 2	28 Pipes 1
3 Patch 3	16 Organ 1	29 Pipes 2
4 Sitar	17 Organ 2	30 Cavern
5 Bowed 1	18 Soft Pad	31 Pipes 3
6 Sci-Fi 1	19 Patch 19	32 Shaky Fifths
7 Sci-Fi 2	20 Patch 20	33 Obligatory Bell
8 Dithered Zither	21 Patch 21	34 Polysquares
9 Patch 9	22 Strings 1	35 Basic subtractive 1
10 Patch 10	23 Bottle Blow	36 Basic subtractive 2
11 Soft Arpeggios	24 Crystals	37 PWM Fifths
12 Lead 1	25 Clavinet	38 Telstar Saw

*Starting with firmware v7, FLASH allows you to select any patch automatically at startup. See **Software – UTE > Startup Patch**.*

*** Also starting with firmware v7, CC 119 can be used to select patches. This was added because some MIDI controllers cannot send patch change messages, but can send CCs.*

Tuning

FLASH has been designed from the ground up with microtonal music-making in mind. To that end, it supports two different forms of microtuning: multi-channel pitch bending, and pre-defined microtonal tuning tables.

Pitch Bend

Multi-Channel Pitch Bend support is also known as *General MIDI microtuning* (which is a subset of *Multidimensional Polyphonic Expression*, or *MPE*). It allows immediate microtonal output from controllers like the Tonal Plexus, or tuning devices like TBX1 or TBX2/b (in POLY mode). When you plug FLASH into the MIDI OUT socket of any of these devices, FLASH produces unlimited microtonal output according to the tunings stored in the other devices, so that there is no need for using the Tuning Tables in the synth.

Standard MIDI Registered Parameter Numbers (RPN) are used to set the Pitch Bend Range on a per-channel basis as follows:

CC 101, value 0

CC 100, value 0

CC 6, with value equal to desired Pitch Bend Range in semitones

The default pitch bend range for all channels is the MIDI standard of +/- 2 semitones (one whole-step up or down). Pitch Bend is calculated internally as a floating point value, respecting the full 14 bits of all Pitch Bend messages. Note that the Pitch Bend control of keyboard may send fewer than 14 bits (check the documentation for your controller). Note also that as of firmware v2, when using a custom tuning table, the Pitch Bend range for that channel is set to a fixed amount which cannot be changed as long as the Tuning Table is being used.

Tuning Tables

If you want microtonal output but you don't have a Tonal Plexus, TBX1 or TBX2/b, then you can use custom Tuning Tables inside FLASH. Tuning Tables are designed and sent to FLASH using UTE software and the FLASH-programmer (see **Software – UTE**).

When the synth receives a MIDI Note message, the normal expectation is to hear the Note sound which is assigned to that MIDI Note number. For example, if you send MIDI Note 60, you expect to hear a middle C, and MIDI Note 72 will be the C an octave above middle C. FLASH lets you change the pitches assigned to the MIDI Notes individually on every channel. Said another way, the sounding pitch output from FLASH depends on the tuning table which has been assigned to the MIDI Channel on which the Note message has been received. By default, FLASH uses standard twelve-tone equal temperament (12ET) tuning tables for all channels, so

the MIDI Note frequencies output are normal 12 tone tuning at A440. But FLASH also stores 16 custom tuning tables – one for each MIDI channel – to allow you to assign any frequency to any note on any channel. This allows you to use any standard MIDI controller to produce microtonal output.

The default tuning 12ET is tuning table 0, and custom tuning tables are numbered beginning with 1. The following 16 default tuning tables are included starting with firmware v2.

	Default Tunings	Comments
0	12 - ET	standard Western tuning at A = 440 Hz
1	5 - Olympos	a Just Intonation pentatonic (rising and falling)
2	7 - Boethius	a historical “blues scale” (rising and falling)
3	9 - AlFarabi	
4	10 - Portuguese	
5	12 - 7LimitJI	
6	12 - Meantone	standard 1/4 syntonic comma
7	12 - Slendro+Pelog	5-tone and 7-tone scales from Java and Bali
8	15 - TonesFrom22ET	
9	19 - 5LimitJI	
10	19 - ET	
11	24 - AlFaribi10thC	
12	31 - ET	(tuning of the Fokker organ in Amsterdam)
13	43 - Chromelodeon	Harry Partch historical reed-organ tuning (in G)
14	53 - ET	tuning of the Bosanquet harmonium
15	72 - ET	
16	96 - ET	

Note that each tuning table consists only of a list of 128 frequencies in Hz. FLASH does not currently store any other information about the tunings in its memory. We may add internal storage of additional data like tuning names in a future update.

There are two ways of selecting tuning tables in real time. The first method works per-channel using CC 3.

0	Set only this MIDI channel's tuning table to 12 ET
1 - 16	Set only this MIDI channel's tuning table to specified table number

So, to use for example the 5th tuning table on MIDI channel 7, send CC 3 on channel 7 with the value 5.

As a convenience for multi-channel controllers like the microzone or Lumatone keyboards, a second method of tuning selection sets all tuning tables at once with each channel assigned to a tuning table in sequential order. This is done using CC 9.

CC 9	All-channels Tuning
0	Set all channels to 12 ET
1-16	1 = Set all channels to table 1, 2 = Set all channels to table 2, etc.
any other value	Set channel 1 to tuning table 1, channel 2 to tuning table 2, etc.

So, to load tuning table 7 on all channels, send CC9 with a value of 7 on any active channel. To load all tuning tables in sequential order, send CC9 with any value above 16.

Tonal Plexus TPX owners tip: if you want to use FLASH's internal tuning tables with a TPX keyboard, set the keyboard's TUNED OUTPUT to OFF. Otherwise you will be sending tuned output to the synth which is already using retuned lookup tables, and the result will be wrong. The easier option is to use GM microtuning TUNED OUTPUT from the Tonal Plexus to retune the default 12ET FLASH tuning.

*Starting with firmware v5, FLASH also allows you to select tuning tables to load automatically at startup. See **Software – UTE > Startup Tunings**.*

Synthesis

Firmware v6 and above features three algorithms, selected and configured by MIDI CCs.

Algorithms

The desired synthesis algorithm is selected via CC 24. Available algorithms have variations differing in numbers of oscillators, polyphony and stereo versus mono audio output. More algorithms may be added in future firmware updates.

CC 24	Description	Polyphony	Output
0	Alg. 1 — FM polysynth	16	mono
1	Alg. 1 — FM dual oscillator synth	8	stereo
2	Alg. 1 — FM monosynth (dual osc)	1	stereo
3	Alg. 1 — FM quad oscillator synth	8	stereo
4	Alg. 2 — Waveshaper polysynth	16	mono
5	Alg. 2 — Waveshaper dual oscillator synth	8	stereo
6	Alg. 2 — Waveshaper monosynth (dual osc)	1	stereo
7	Alg. 2 — Waveshaper quad oscillator synth	8	stereo
8	Alg. 3 — Band-limited polysynth	16	mono
9	Alg. 3 — Band-limited monosynth (triple osc)	1	stereo

Any other value for CC 24 will select the default Algorithm 1 with polyphony 16, mono output.

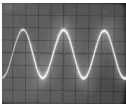
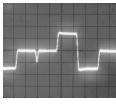
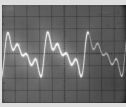
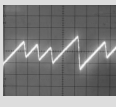
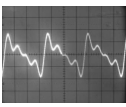
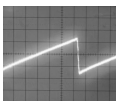

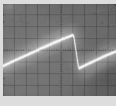
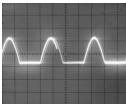
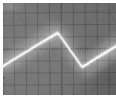
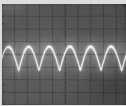
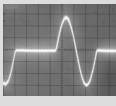
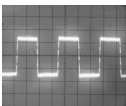
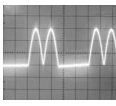
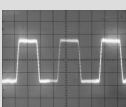
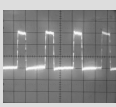
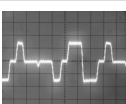

When changing the algorithm, all currently playing notes will be stopped immediately, so you may want to lift all keys and allow the notes to release before changing the algorithm.

Tip: To avoid abruptly stopped notes, lift all notes before changing algorithms.

The specifics of each algorithm are given below, after an introduction to the main elements which constitute the algorithms.

Waveforms

Algorithms 1 and 2 use wavetables, and CC 25 loads a waveform which acts as an operator. Some waveforms additionally allow an extra parameter via CC 26, as indicated below.

CC 25			CC 25		
0	Sine		9	Octaves square wave	
1	Hammondish 1*		10	Fifths Sawtooth wave	
2	Hammondish 2*		11	Hard Sawtooth wave	
3	Sine Cubed		12	Soft sawtooth wave	
4	Half Sine		13	Variable Sawtooth wave **	
5	Sine absolute		14	Sine, even periods only	
6	Hard square wave		15	Sine absolute, even periods only	
7	Soft square wave		16	Hard pulse ***	
8	Fifths square wave		17	Soft pulse ***	

* CC26 = binary permutation of 7 “Drawbars” | ** CC26 = slope | *** CC26 = duty cycle

Any other value sent for CC 25 will select the default Sine waveform. More waveforms may be added in future firmware updates.

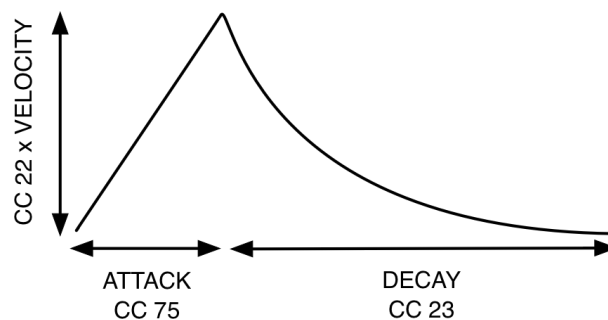
For Algorithm 3, CC 25 selects the behaviour of oscillator 3 as explained below under the heading **Algorithm 3**.

When experimenting with different sounds, changing the waveform while playing can be helpful, but any voices currently sounding will have a discontinuity as the waveform is loaded.

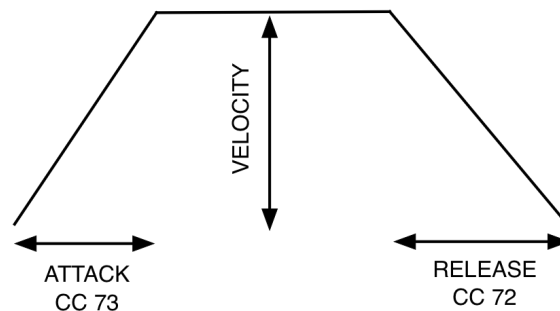
Tip: To avoid “broken-wave” output, lift all notes before changing waveforms.

Envelopes

Two envelopes are available to various algorithms. The first envelope normally controls the Effect Depth (in the case of Algorithm 1, the FM depth). The adjustable parameters are Attack Rate, Decay Rate, and Effect Depth. The Decay is exponential, at its longest it does not decay at all and allows for fixed Effect Depth. The envelope height is a function of both CC 22 and note velocity.



The second envelope is used to control the overall voice output according to note velocity with two adjustable parameters: Attack and Release.



Note that for reasons of efficiency and processing optimisation many parameters of the envelopes are cached (such as FM Depth) such that sending CC values will not affect notes already sounding. To experiment with different values for the envelopes, it will be helpful to play a note repeatedly as you adjust the parameters instead of holding sustained notes.

Tip: Play repeated notes when adjusting envelopes to hear changes in real time.

Low Frequency Oscillator (LFO) and Noise Source

The mod wheel (CC 1, Modulation) controls the depth of the LFO, which is applied to note pitch (vibrato). The LFO frequency can be adjusted on CC 76. When CC 76 is set to 127, the LFO is disconnected and replaced with a sampled noise source, which again can be controlled by the mod wheel.

Detune

All modes with multiple oscillators may have some oscillators detuned (chorus effect) using CC 15. This parameter is not intended for exact microtonal control. The detune amount at maximum value of 127 puts the oscillators about 43 cents apart. For dual- and triple-oscillator algorithms, one oscillator is bent up and the other is bent down. For the quad oscillators, they are all spread equally. This allows the parameter to be controlled by a single CC value. More advanced detuning control may be added in a future firmware update.

Portamento and Arpeggiator

Portamento and arpeggiator are available only in monophonic mode. Portamento is adjusted with CC 5, and the arpeggiator speed is set using CC 27, where 0 is fastest and 127 is stopped. Note that very fast arpeggiator settings will result in noisy sounds with no discernible pitch centre when holding down multiple keys.

Output Gain (advanced)

There is a final gain adjustment exposed, for a couple of reasons:

1. Because of the nature of a powered-by-MIDI device, the output is fairly quiet.
2. Small amplitude outputs may induce artefacts which an output gain increase can correct.

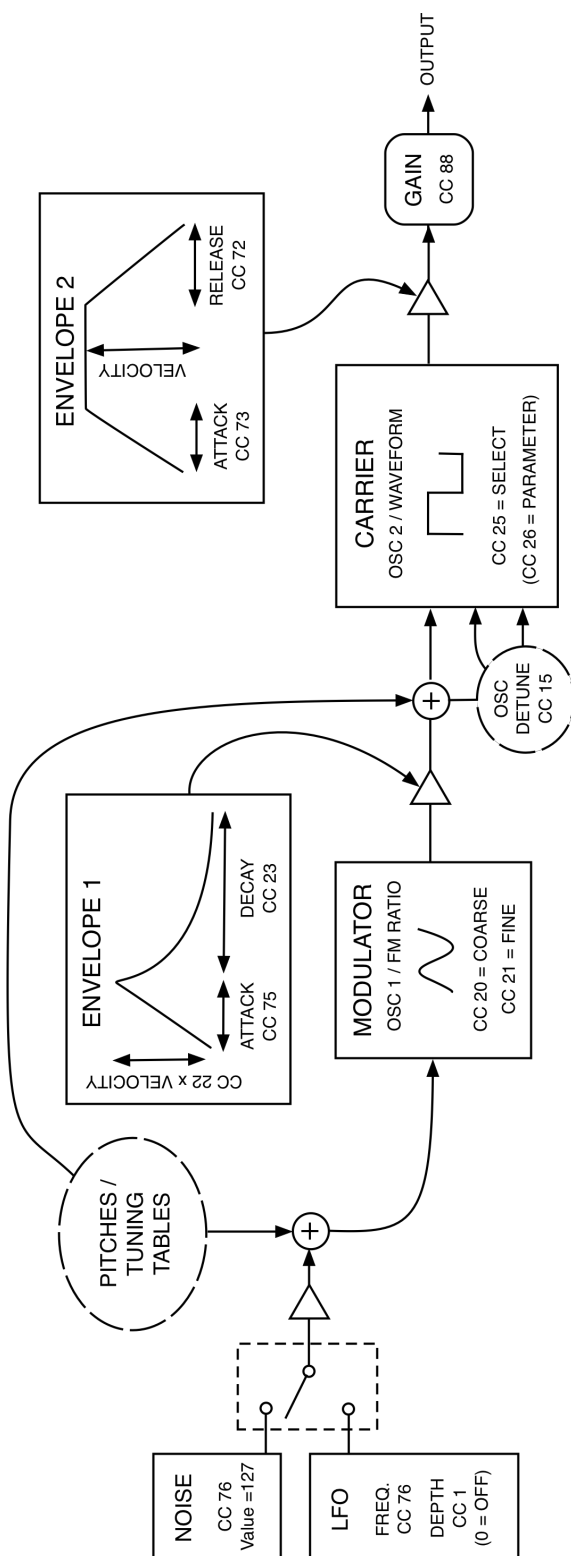
Output gain is set by CC 88. A value of zero corresponds to unity gain, and higher values increase the gain. Note that due to the caching nature of the envelopes, gain changes will not correctly take effect on notes currently playing. For patches with a polyphony of 8, a safe maximum gain value is 16. For monophonic patches, all gain values are considered safe.

Warning: *If you turn up the output gain, you may induce integer overflow, which sounds horrible (crackling, clipping, etc.) Use at your own risk!*

Algorithm 1

The default algorithm is straight two-operator FM synthesis, technically phase modulation (not unlike the Yamaha DX7). The modulator is a sine wave, and the carrier can be customised with a number of different waveforms. Dual- and quad-oscillator versions of algorithm 1 add

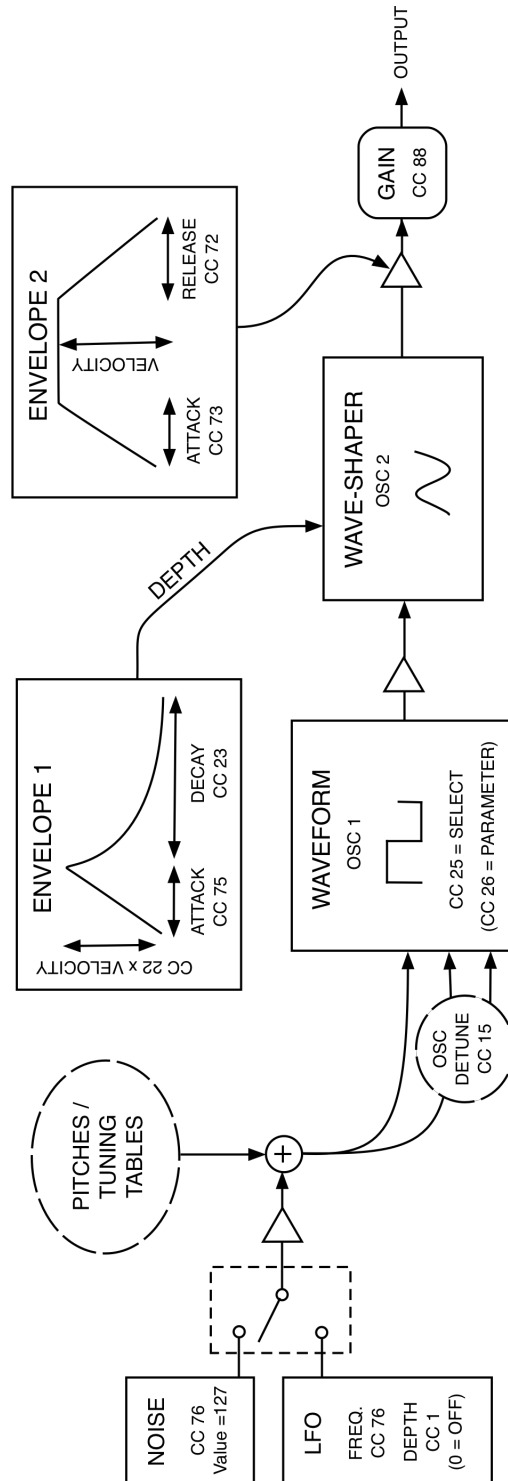
extra carrier waveforms with adjustable detuning. The FM Ratio is controlled by CC 20 (coarse adjustment) and CC 21 (fine adjustment). The LFO can be optionally replaced with a noise source to inject “chaos” into the sound. A diagram is given below.



ALGORITHM 1: Phase modulation two-operator FM synthesis

Algorithm 2

This (non-FM) algorithm is similar to the first, but with oscillator roles swapped. Oscillator 2 is used as a wave-shaper, where Envelope 1 controls wave-shaping depth. The LFO / Noise option is the same as in Algorithm 1. Note that if the effect depth (CC 22) is set to zero, no sound will be produced. A diagram is given below.



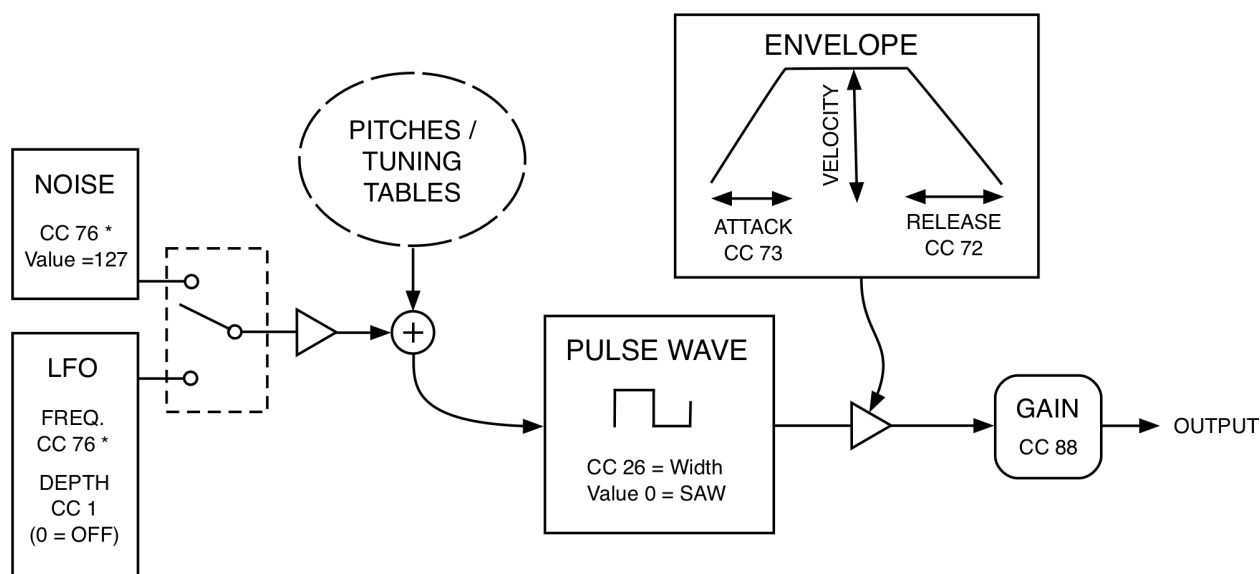
ALGORITHM 2: Wave-shaping synthesis

Algorithm 3

Our third algorithm addresses a few shortcomings in the first two algorithms, and also provides some room for expansion in future updates. Although the first two algorithms offer square and sawtooth waves as the carrier or the primary waveform, these waveforms tend to sound acceptable only in the midrange, with low notes sounding a bit dull and high notes sounding rather less than beautiful. The reason for this is that FM synthesis requires soft-edged waveforms, which square and sawtooth waves by definition are not. The envelopes used in algorithms one and two also have a small quirk, in that they never reach an attack/release time of exactly zero, for similar reasons. Our third algorithm addresses these shortcomings, providing true square/pulse and sawtooth waves which are very clean and crisp throughout the whole frequency range, with envelopes allowing zero-lengths. The monophonic form of this algorithm makes use of band-limited steps, which are called “BLEPs” in academic papers. More specifically, it uses polynomial BLEPs or “PolyBLEPs”. Although these are admittedly exciting names, we prefer to avoid easily misconstrued jargon, so we refer to the algorithm merely as “band-limited”.

Band-limited Polysynth

Clean waveforms with standard envelope, LFO and standard modulation control currently constitute the sum total current offering of Algorithm 3 in its 16-voice poly form (CC24 = 8), as additional processing for this many voices using this algorithm has so far proven a bit problematic for MIDI-power. Luckily the high quality of the pulse / saw waveforms alone provide ample reason to make polyphonic music with this algorithm. A diagram of this simple polysynth is given below.



ALGORITHM 3 — POLY: 16-Voice Pulse-wave synthesis

Polysynth Controls

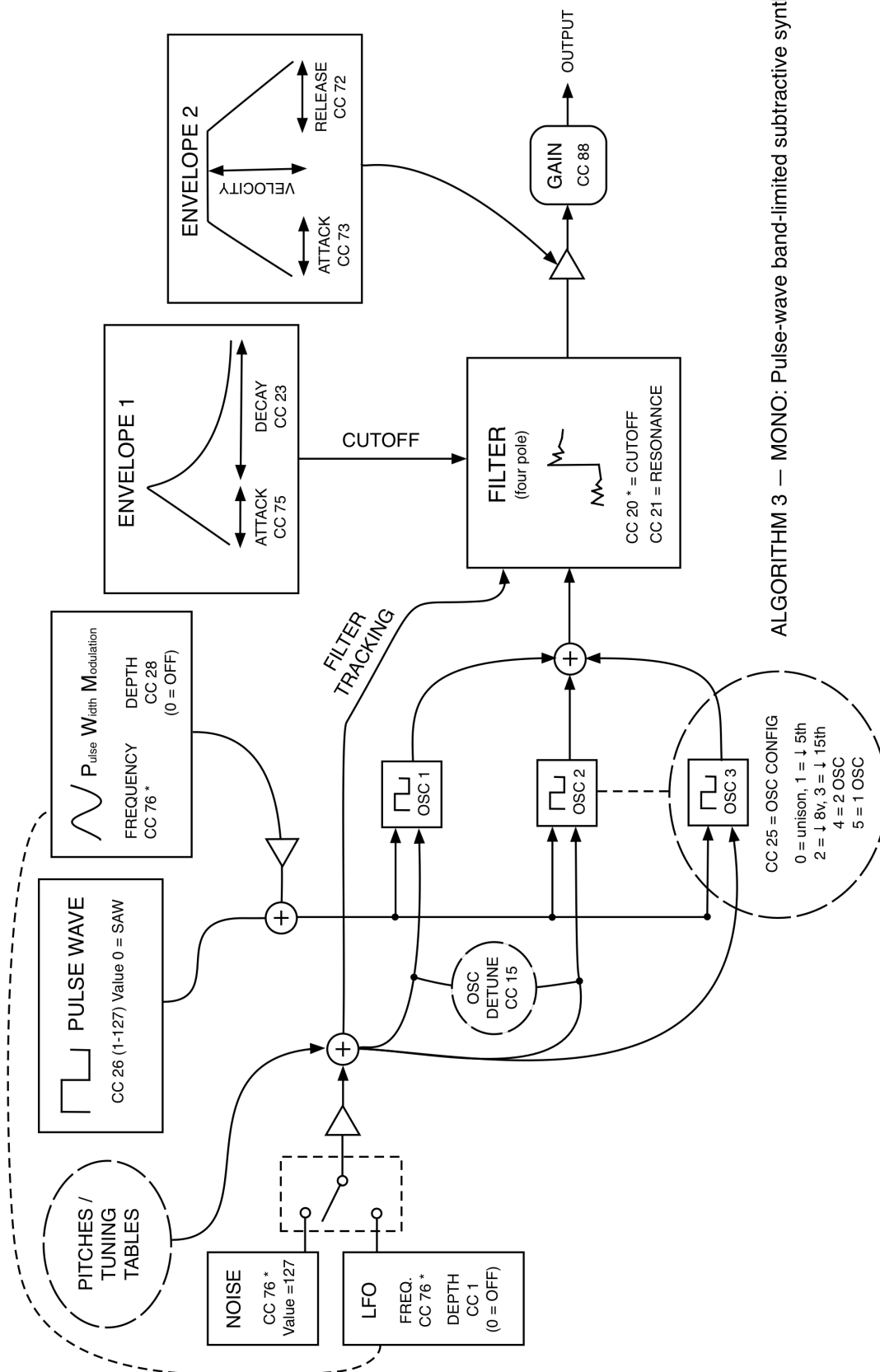
Here you may notice the absence of Envelope 1 and effect controls. Envelope 2 controls are the same as Algorithms 1 and 2. "Wave parameter" (CC26) controls the duty cycle of the waveform, or switches it to sawtooth with a value of 0. LFO and modulation remain the same.

Band-limited Monosynth (pseudo-moog)

The monophonic version of our algorithm (CC24 = 9) brings out the full potential of our synth, and is much more involved than any of the previous algorithms, combining three oscillators with a pitch-tracking four pole resonant filter to make an efficient monophonic subtractive synth, with filter cutoff that can be altered smoothly in realtime. This algorithm is meant to loosely approximate the original *minimoog* synthesizer. The currently limited number of oscillator configurations (CC25) are based on diagrams in the original *minimoog* patch book. More variations may be added in future firmware updates.

Monosynth Controls

The monophonic synth CCs share some core things in common as the first two algorithms so that switching between algorithms on the fly remains fairly intuitive, though there are some important differences to keep in mind, as follows. The FM frequency knobs now control Filter Cutoff (CC20) and Filter Resonance (CC21). Note that if Filter Cutoff is set to zero, no sound will be heard. Envelope 2 controls remain the same, with Envelope 1 now applying to the Filter, but without the Depth parameter, since this is handled by Cutoff and Resonance. For a sustained sound, set the Decay parameter of Envelope 1 at or near to zero. Higher values will result in more quickly decaying increasingly percussive sounds. The Wave Parameter (CC26) controls the "Duty Cycle" (pulse width) of the waveform, or switches it to sawtooth with a value of 0. "Detune" alters the first two oscillators (for details, see under the heading *Detune* above). Since this is not a wavetable algorithm, the selection of waveforms used in algorithms 1 and 2 are not available. Instead, "Waveform Select" (CC25) becomes "OSC Config" and controls the behaviour of our three oscillators. Values 0 through 3 apply to the third oscillator only, changing its behaviour as follows: 0 = unison, 1 = down a 5th, 2 = down an octave, and 3 = down two octaves from the other two oscillators. Values 4 and 5 optionally turn off 1 or 2 of the oscillators for a leaner sound: 4 = third oscillator off, 5 = oscillators 2 and 3 both off. Arpeggiator, LFO rate, and portamento behave as expected; however, a significant difference to the other algorithms is that the LFO rate applies to not just one but *two* kinds of modulation, normal LFO and *Pulse Width Modulation* (PWM). The latter gives a sweeping or flanging effect, as it changes the duty cycle according to the LFO frequency. For the uninitiated, using PWM and Modulation together can be a bit tricky at first, with results that can be quite subtle. We recommend first setting the PWM control to zero, then adjusting the Modulation control. Conversely, to try out PWM, first set the Modulation control to zero. Once you get used to controlling each independently, mixing them will be easier. A diagram of the monosynth is on the following page.

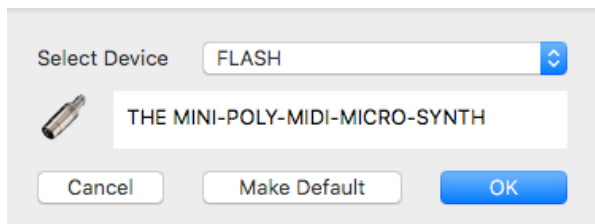


ALGORITHM 3 — MONO: Pulse-wave band-limited subtractive synthesis

Software – UTE

This section explains how to use Universal Tuning Editor with FLASH. The interface itself should be fairly self-explanatory, but the following information will help you use it.

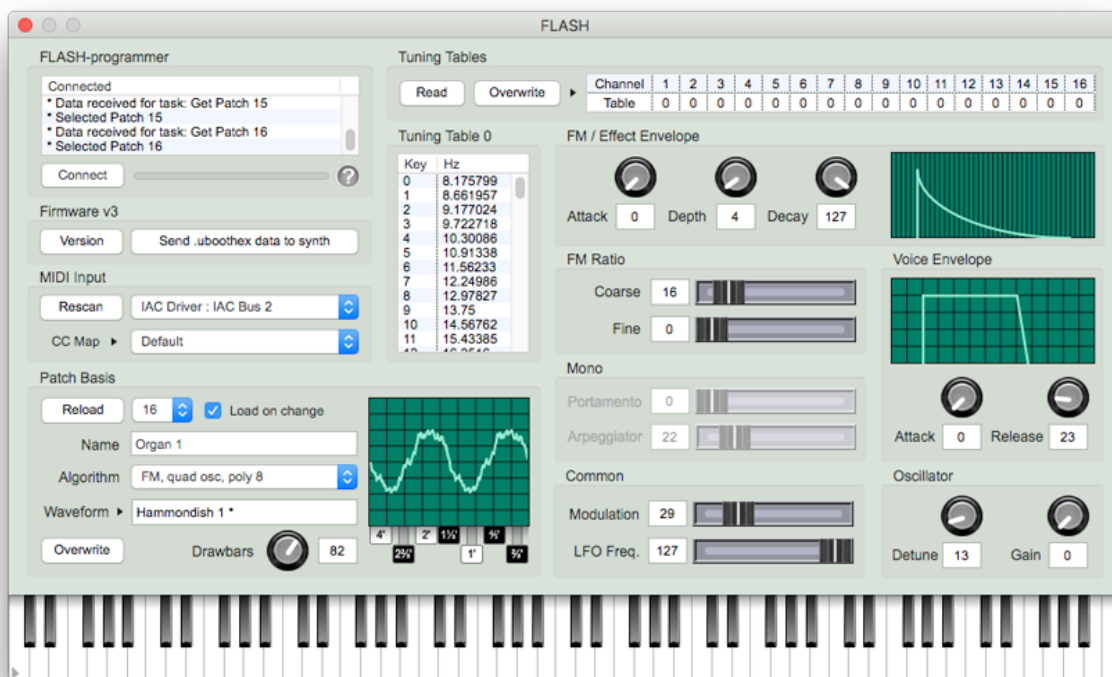
Select FLASH under Devices



The first step for using FLASH with UTE is to select FLASH under the **Devices** toolbar item. This toolbar item may be named with the selected device (e.g. **TBX2**). Choose **Select Other Device** from the popup menu. You can make FLASH the default device by clicking the button **Make Default** before clicking **OK**.

Open the FLASH Window

After FLASH is selected as the Device, click the **Devices** toolbar item again and select the menu item **Open FLASH Window**.

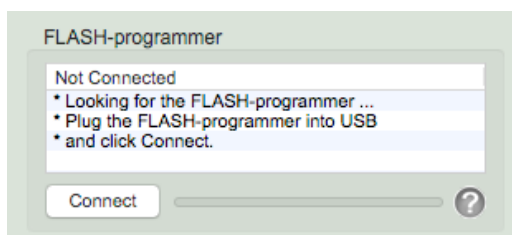


The interface is organised roughly from top to bottom and left to right in the order you need to set things up to use it.

Bootloader vs. MIDI Function

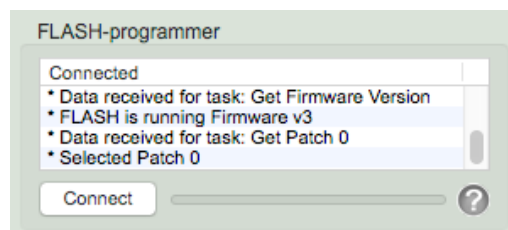
The interface allows you to work with the synth normally in real time using the on-screen controls or your MIDI controller. It also allows you to update firmware and read and write tunings and patches. These tasks take place in two fundamentally different ways. All Programming takes place through the *Bootloader* of the chip which is running the synth firmware. The bootloader is written by STM (the company who makes the chip). All synth output takes place while running our firmware, which is written by mitxela. MIDI and Bootloader functions are mutually exclusive. In other words, while the bootloader is running, there is no synth function, and while the synth is running, there is no bootloader function. The software switches between these functions automatically depending on what you choose to do.

Connecting the Programmer

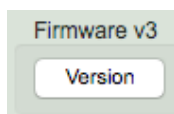


The top left corner shows the status of the FLASH-programmer. If the programmer is not connected, follow the displayed instructions. If the programmer is connected but does not work properly, click the question mark icon and follow the directions.

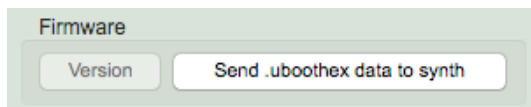
When you plug in the programmer, its LED will blink twice, then remain unlit (the LED only lights up when data is transferred). Once the programmer is connected, the display should look something like the image at right.



Firmware Version & Updating



Immediately after the programmer has connected and the synth has been initialised, the firmware version will be read from the synth, the server online will be checked for a possible available firmware update, and any special notices about the firmware version will be displayed. Once the firmware version has been read, it appears beside the word **Firmware** in this area. After this step has been completed, synth interface is ready to use.



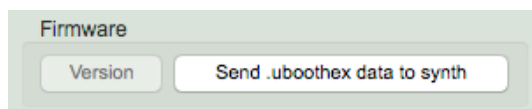
A *.uboothex* file contains an encoded firmware data for H-Pi Instruments devices. In this case, our file includes the firmware, default patches, and default

tunings. When you send a firmware update, you may not want to overwrite patch and tuning data. Or, you may want to reinstall the default settings for patches or tunings but not

overwrite the firmware. This is why the button ***Send .uboothex data to synth*** is so named, (rather than ***Firmware Update***). Clicking this button opens a popup menu with several options. After making your selection, you must navigate to the .uboothex file from which you want to install data. Links to .uboothex files are found on the FLASH webpage. The selected operation is then executed, and progress is shown in the FLASH-programmer display area.

Open Source Firmware

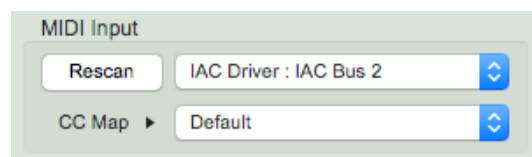
Since August 2022, FLASH firmware has been made open source, so that anyone can make their own version of the firmware to suit their needs if they are so inclined (see the product page for the Github link). *We do not provide any technical support for user custom firmware programming or compiling!*



If you have successfully programmed and compiled a custom .bin firmware image for FLASH, and you have confirmed that your firmware functions properly, UTE provides a way to convert your .bin

file into a .uboothex file so that you can potentially share your firmware with other users. Click the button marked ***Send .uboothex data to synth*** and select the option ***Convert .bin to .uboothex***. The resulting file can then be loaded by UTE for firmware updates by any user. We will maintain an archive of custom firmware for FLASH. To have your firmware considered for addition to the archive, send your .uboothex file to us as an email attachment, explaining what enhancements you have added to the synth.

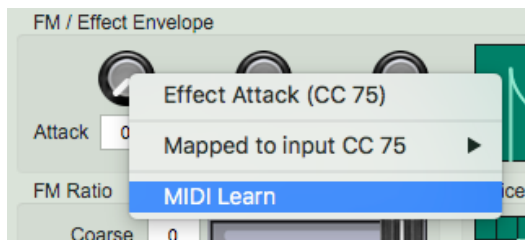
MIDI Input, CC Map, & MIDI Learn



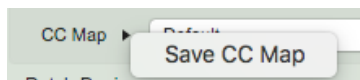
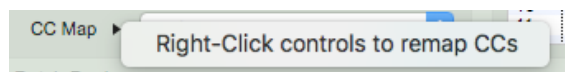
Select your MIDI device for input. If you have multiple controllers you would like to combine for input, you must currently either merge them via MIDI before sending the signal to the computer, or use an external MIDI routing app such as ***MIDI Patch***

Bay on Mac or other MIDI routing software on Windows to merge the signals before sending them to UTE.

If you are using physical sliders and knobs to control CC messages, when the knob or slider is sending a CC number corresponding to a FLASH control parameter, the on-screen control will react and move along with the physical control. If your physical controller is sending some other CC, you can still link it to an on-screen control using either a menu selection, or the ***MIDI Learn*** option. ***Right-click*** on any on-screen knob or slider to open a popup menu. This menu displays the CC number which is output from this control, an optional alternate CC mapping assigned to the control, and an option for ***MIDI Learn***.

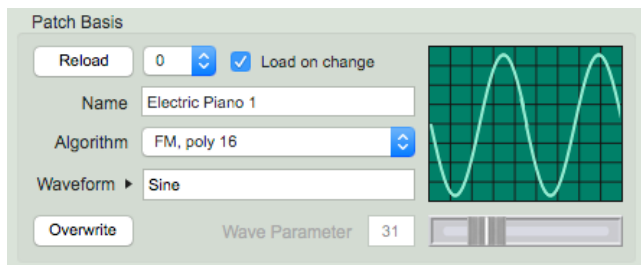


In the case displayed above, the control sends CC 75 to the synth. To control this with some physical knob or slider which sends some other CC, you can click the **Mapped to CC 75** and select the other number from the submenu, or you can select the menu item **MIDI Learn**, and then move the physical control, which will then be linked to the on-screen control. If you choose to remap any CCs, you should then save the configuration as a CC Map. Click the triangle to the right of **CC Map** to open the popup menu. If no CCs have been remapped, you will only see an instruction to right-click controls to remap them.



If you have remapped them, the option to save the CC Map appears. Maps are saved with the name of the selected MIDI input. Multiple maps using the same input are numbered.

Patch Basis



The **Patch Basis** area contains controls for selecting patches by number, displaying and editing patch names, selecting Algorithms and Waveforms, and reloading or overwriting a patch.

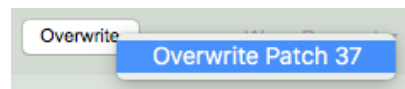
For patches which implement a Waveform having a **Wave Parameter** control, a slider or drawbars interface becomes active. The static display (which does not reflect real-time synth output) shows two cycles of the selected waveform.

Editing a Patch & storing that as a new Patch

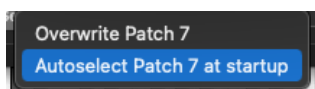
The **Load on change** checkbox allows you to control whether patches will be loaded from the synth when the patch number control is used. Use this control to read a patch, make changes, and save it as a new patch to an unused location. Here are the steps to follow:

1. Load a patch by selecting a number from the popup (Firmware v3 includes default patches for numbers 0-33) (**Load on change** must be checked for the patch to load).
2. Make any changes you want to the patch (change Algorithm, waveform, any knobs or sliders, etc.)

3. Uncheck **Load on change**.
4. Using the popup menu, select the number of an unused patch you want to program. (Here is where the purpose of the checkbox should be clear. If you had not unchecked **Load on change**, then the patch number you just selected would be loaded from the synth, and your altered patch would be lost.)
5. Click the “**Write / Set**” (formerly “**Overwrite**”) button. To prevent you from accidentally clicking this and overwriting a patch you did not actually intend to overwrite, a popup menu appears. Select the menu item to overwrite the patch (for example a currently empty Patch 37).
6. Check **Load on change**.



Startup Patch

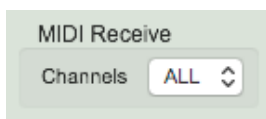


By default FLASH starts up with patch 0 selected. As of FLASH firmware v7 and UTE 2.3.6, you can change this behaviour by following these simple steps:

1. Select the patch you want to assign as the Startup Patch.
2. Click the “**Write / Set**” button, and select the menu option “Autoselect Patch at Startup”.

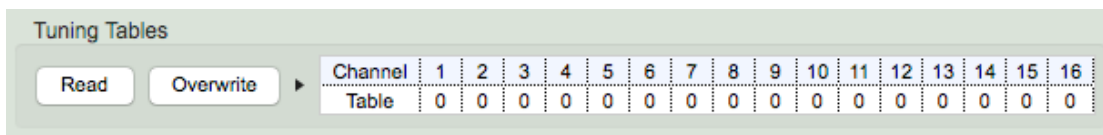
That’s it. The next time you plug in the synth, it will start with your desired patch already selected.

MIDI Receive Channel(s)



By default FLASH responds to all MIDI channels, but this may not be optimal for every situation. As of FLASH firmware v4 and UTE version 1.6.0 (you must have both or higher to have this feature), you can select the channel on which you want the synth to receive using the popup menu. The value is stored in FLASH and remembered when you plug in the synth for normal use.

Tuning Tables



Assign tuning tables to MIDI channels by clicking the triangle and selecting options from the popup menu. Note that tuning tables cannot be assigned to MIDI channels on which the synth is not receiving (so to use this feature in the editor, you should first set the receive channel to ALL). *Note also that these selections are done by MIDI CC and are not automatically stored.* To store these selections, see **Startup Tunings** below.

Troubleshooting

You may run into problems with using FLASH. Here are issues we know about.

1. ***There is sound at all*** — If this happens when you plug it into a MIDI OUT socket, that means the host socket is not providing enough power. This may happen even if you have measured the power output from the socket and confirmed that it is within the expected range (see ***Precautions***). Usually connecting a powered MIDI THRU box or connecting THRU from another MIDI device and plugging the synth into that will supply the needed power.
2. ***I hear a buzzing or hissing sound*** — In this case, you need to add a *Ground Loop Isolator* on the audio output line from the synth.
3. ***I hear a sort of high-pass white noise, especially when I play low notes*** — This is a known issue with some patch configurations, and is a limitation due to very low power consumption. Nothing can be done about it, other than to try to tweak the patch until the problem is minimised.
4. ***I hear a soft “click” at the onset of notes when I play*** — Some patch configurations may cause this. Simply tweak the patch in the UTE editor until the problem is minimised or goes away.
5. ***When connected to the programmer and running UTE, output is distorting when I play more than a few notes at a time*** — This problem only affects owners of first-run FLASH programmers sold in 2019; the problem has been solved with a hardware change to our programmers starting with our second production run. To owners of the first run programmers: we know what causes the problem, but regrettably there is no way for us to fix it in our firmware. The problem is due to a bug in the GO command of the STM bootloader, over which we have no control. When connecting the synth to a first-run programmer, output to the synth must be limited to one or two notes at a time in order to avoid distortion. *All programmers made after the first run do not have this problem!*
6. ***The programmer doesn’t seem to work on my Mac*** — Almost all Macs will not need the FTDI driver, but on some older machines you will need to install the driver. Download it from the product page at <https://hpi.zentral.zone/flash>
7. ***I installed the FTDI driver on my Mac and now the programmer doesn’t work*** — Follow the instructions below to remove the FTDI driver from your computer:
[1] In Terminal, type: `sudo rm -rf /System/Library/Extensions/FTDIUSBSerialDriver.kext`
[2] Enter your system password. [3] Restart your system.

8. *(On Windows) When I try to program the firmware, it stops at some page and UTE stops responding* – This problem has been reported by a few customers, who describe it as intermittent and not exactly reproducible. A Windows computer may apparently do this because of differences in underlying hardware and drivers for the serial traffic on the system. Since we cannot reproduce the problem, we can't do anything about it yet, so the only solution is to try again until it works. If UTE stops responding, you have to use **Task Manager** to **End Task** and then reboot UTE to try again. If you see **Access Denied** when you try to do that, then you need to grant access to the application under Windows security.
9. *I'm experiencing some other issue* – In most cases this is a matter of understanding how the synth works. In other cases a firmware update might solve the problem. Please first make sure you have read through this documentation completely. If the problem remains, then we need to correspond with you to figure out what might be causing it. Send an email to hpiinstruments@zentral.zone or zentralzone.alert@gmail.com

Appendix

The information in this section is added at customer request.

CC26 and the “Hammondish” Waveform Drawbars

In order to simplify the control to a single value, the seven data bits of the MIDI value byte for CC26 are interpreted as seven on/off “drawbars” which add upper harmonics to the fundamental waveform. The harmonic numbers, “pipe” lengths, and respective amplitudes of each added wave corresponding to each bit are shown in the following table.

Bit Position	Harmonic Number	“Pipe” Length	Hammondish 1 Amplitudes	Hammondish 2 Amplitudes
1	2	4'	1/4	1/4
2	3	2 $\frac{2}{3}$ '	1/4	1/4
3	4	2'	1/4	1/5
4	6	1 $\frac{1}{3}$ '	1/4	3/20
5	8	1'	1/4	3/20
6	10	$\frac{4}{5}$ '	1/4	1/10
7	12	$\frac{2}{3}$ '	1/4	1/10

As the CC value increases from 0 to 127, higher harmonics are added according to binary counting logic. Combinations may be found by writing out binary values and converting them to decimal or vice versa. For example, if you want to add harmonics 4 and 10, this corresponds to bit positions 3 and 6. Positions are counted from right to left, so the value is 0100100 = 68. Or, converting in the opposite direction, the value 113 in binary is 1110001, so for this value bits 1, 5, 6, and 7 would be turned on, adding harmonics 2, 8, and 12. Since finding desired combinations in this manner may be for normal humans somewhat nonintuitive, the software (UTE) provides an interface for engaging and disengaging each “drawbar” individually. Simply click on drawbars to turn them on or off, and a CC knob will jump to the position of the corresponding value, sending the correct data byte to the synth.

Previous Versions Change Log

Here are lists of changes for previous versions of this documentation. Changes for the current version are found in the section *Change Log*.

v7 - August 10, 2022

- Added **WARNING** and edited text for clarity under *Precautions - Power Requirements*
- Added information about the LED behaviour of the programmer under *Introduction – FLASH Programmer* and under *Software - UTE – Connecting the Programmer*
- Added topic *Open Source Firmware* under *Software - UTE*
- Added information to *Troubleshooting* about programmer driver issues on Mac.

v6 - November 27, 2020

- Updated *Introduction – History*
- Changed language “any channel” to “any active channel” since starting with firmware v4 the synth can be set to respond only on a single channel, or on all channels.
- Reorganised and updated *MIDI – Continuous Controllers* table for better cross-referencing with software and the Algorithm diagrams.
- Reorganised and partially rewrote the chapter on *Synthesis* with added *Algorithm 3*.
- Updated diagrams of Algorithms 1 and 2 and added two new diagrams for *Algorithm 3*.
- Updated *MIDI – Program Changes* with new patches introduced in firmware v6.

v5 - October 30, 2020

- Added notes about setting startup tuning tables in *Tunings – Tuning Tables* and in *Software – Startup Tunings*.
- Changed this documentation version numbering to whole numbers to match the synth’s firmware version numbering (firmware v5 matches documentation v5 instead of v1.5).

v1.4 - March 23, 2020

- Added notes about setting the MIDI receive channel under *MIDI – MIDI Channels* and in *Software – MIDI Receive Channel(s)*.

v1.3 - January 20, 2020

- Explained selecting FLASH as the device a bit more clearly in *Software – UTE* and fixed some typos.
- Fixed location of firmware updating instructions under *FLASH-programmer*.
- Added a reported issue using UTE on Windows in the *Troubleshooting* section.

v1.2 - January 15, 2020

- Added chapter *Software – UTE* explaining how to use Universal Tuning Editor with the synth and programmer to read and write patches and tunings and to update firmware.
- Improved some wording in the *Troubleshooting* section.

v1.1 - January 13, 2020

- *FLASH Firmware v3* has been released – fixed MIDI Clock problem, added MIDI System Reset, enhanced behaviour of CC 9 (All-channels Tuning select).
- Added section on *Troubleshooting*
- Added missing label for CC 75 to FM diagram
- Added missing details to oscillator *Detune* parameter listing
- Fixed listing of *Output Gain* default unity value (it is zero, not 8)
- Added diagram for synthesis *Algorithm 2*
- Updated diagram for CC9 behaviour for *Firmware v3*
- Clarified binary permutation for CC26 when using the *Hammondish* waveforms
- Added *Appendix CC26 and the “Hammondish” Waveform Drawbars*

v0.9 - December 1, 2019

- First release, diagram of *Algorithm 2* and some details not yet included

Credits

FLASH Synth and FLASH-Programmer concept / physical design by AAH. FLASH Synth PCB design & firmware by mitxela (maintained since 2023 by Henry Lowengard). UART FLASH-Programmer PCB & firmware by JDP († 2020).

This documentation is written by AAH with contributions from mitxela and Henry Lowengard.

Thank you for supporting H-Pi Instruments and FLASH.

©2019-2023 H-Pi Instruments · FOR THE FUTURE OF MUSIC