XFM2 User Manual

Contents

Thank you for building XFM2!

XFM2 is a “classic” FM synthesizer module built with the technology of the XXI century.

Essentially based on a powerful FPGA chip, its massive parallelism allows for an outstanding polyphony, with a complex dual sound generation architecture, plus two complete sound effect processors, each with up to seven effects running simultaneously.

The power efficiency of XFM2 core processors results in many hours of operation using standard batteries or USB powerbanks. This, combined with its small size, makes it ideal for portable sound and music creation.

XFM2 is light, small, compact, fast, inexpensive, simple to build and use. It has no latency and boots instantaneously. It responds expressively to most common performance gestures and MIDI messages, such as note velocity, pitch bend, aftertouch, modulation wheel, breath control, foot controller, volume, pan, sustain switch and chorus and reverb levels. It’s incredibly powerful and capable of creating very colorful, rich, and expressive sounds.

However, it is a complex creation. Please read through this User Manual to ensure you get the most out of your new device!

As XFM2 is an ongoing project, make sure to regularly check *futur3soundz.com* for enhancements, updates, and expansions.

What is an FPGA anyways?

Back in the 80's, the world had a rapid flooding of electronic devices. Pocket calculators, portable music players (cassette), watches, microcomputers, and numerous other devices which had been previously only imagined suddenly become a reality. Advances in the semiconductor technology, particularly in the *integrated circuit*, combined with novel mass production techniques and industrial automation led this fast-paced development which become the *golden age* of electronics.

One segment of those semiconductors presented particularly notable kinetics, faster than any other: *digital integrated circuits* (ICs). At that time, the technology required to industrially produce ICs was mature enough as to make them accessible at the consumer market. While invented in the late 50's, they were limited to military and government uses for a while, particularly to the Apollo missions.

The best-selling CMOS digital integrated circuit series in history, the CD 4000-4500 series, was released in 1968. As a collection, they provided a complete toolset to design and build any homebrew digital circuit. After several years in which their adoption was hindered by performance issues derived from fabrication process, in the late 70's, almost anyone could purchase a *4-nand gate* IC for sub-dollar prices.

Designers started to design all kinds of digital circuitry, increasingly complex. While discrete, mid and large-scale of integration ICs (up to 10,000 transistors) were sufficient for prototyping and development, large product series required the "etching" of the final design into an *Application Specific IC* (ASIC) in order to achieve maximum cost-efficiency.

The ASIC were (and still are) the perfect electronic solution in terms of speed, energy efficiency, space, reliability, and cost when there's a *fixed* problem to solve, and the number of units to produce is very large. Creating an ASIC can easily be a multimillion-dollar process, so if you want to have one in your product, be sure you'll sell it in the tenths of thousands to offset the costs.

And, make sure you got your design right, as you won't be able to update or change it once it's in the field. With microprocessors, it is possible to update the *firmware* on it, and then fix or expand some functionality. But the hardware itself is fixed forever, so the field update and fix scope are limited.
It didn't take long to the industry to identify the need for a digital circuit that *could* be completely updated in the field, at a *hardware* level: *imagine having a device with a huge number of digital gates, which you could freely interconnect to create complex digital circuit designs, and where every connection would be specified by a software, which you (or your customer) could update anytime, even after years of shipping the final product.*

That device is an *FPGA* (*Field Programmable Gate Array).* In 1985, *Xilinx* invented the first commercially viable FPGA, the XC2064, which presented the world a new technology and market. The 2064 had 64 *configurable logic blocks* to create your digital designs.

Xilinx did never stop growing since then and today, 35 years later, XFM2 is using one of their FPGAs—the *Artix A7*—with over 33,000 configurable logic blocks. The concept of having *gates* as a complexity measurement unit is long gone, as higher conceptual level abstractions serve better to represent design complexity (such as *lookup tables* and *flip-flops).* The FPGAs also started featuring *hardened* elements (easily utilizable fragments of ASIC), such as memories and DSP blocks, many years ago, to increase performance and efficiency.

So, how FPGA compare against other alternatives to create digital circuits, such as MPUs (microprocessor units), MCU/SoC (microcontroller units, system-on-chip) or DSP processors? It's a tricky question. Essentially, it depends on what you build into the FPGA. As it's just a "free digital canvas", one could easily create a microprocessor inside the FPGA. Or many: XFM2 has three microprocessors running together to hold all communication in/out. The advantage starts materializing when you design a perfect digital component for each block in your device.

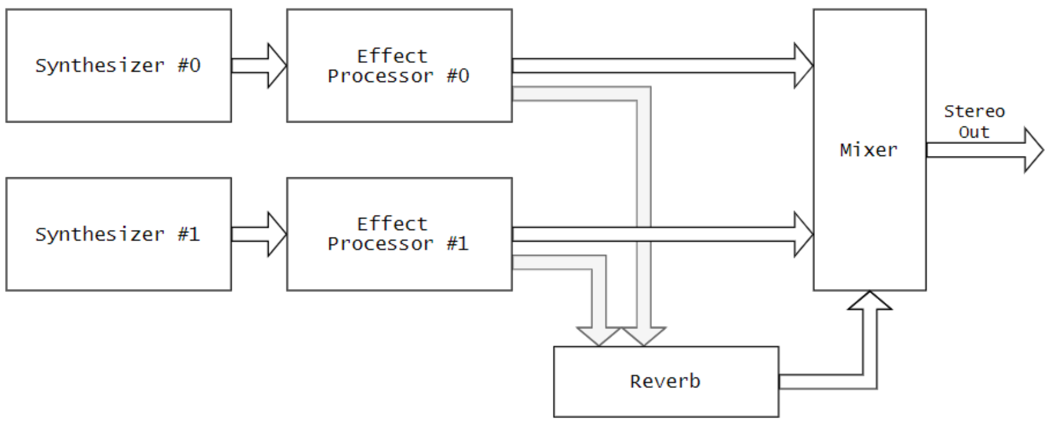
So, in XFM2, a few hundreds of design blocks have been crafted and optimized *down to a gate level*. There are single-core, and hundred-and-twenty-eight-core components running in parallel, depending on the particular needs of the block. Everything has been designed to run *simultaneously* (instead of sequentially, as it is the case in any of the other solutions). The "hot spots" in the design are then massively parallelized.

And still, any part of the circuit could be updated in the future. In fact, the update could be as deep as a completely different synthesizer (or anything which has a MIDI input and an audio output).

XFM2 Architecture

XFM2 is composed by two completely independent 32-voice polyphonic synthesizers, each with its own individual effect processor. Along this manual, each one of the two synthesizer or effect processor will be referred as a *unit.*

This dual-unit architecture allows XFM2 to run as a two-part synthesizer, with up to 64-voice polyphony. A global reverb unit with individual sends for both units completes the XFM2 design.



Voice architecture

Each synthesizer unit can play up to 32 simultaneous FM voices. Each voice features six FM operators, and each operator has two individual oscillators.

As in standard FM synthesis, each operator can play a role as a *carrier*, as a *modulator*, or both simultaneously. Sounds in XFM2 are produced by interconnecting operators with different roles, in different ways. The diagram of those interconnections is usually referred to as an *algorithm*.

Typically, FM synthesizers offer a limited number of algorithms, with fixed interconnection lines.
As an example, the iconic *Yamaha DX7* (the world's most famous FM synthesizer, and one of the best-selling synthesizers in history), featured 32 algorithms, or fixed ways to interconnect operators.

In the DX7, all algorithms had only one feedback loop, and only two of them had the feedback loop routed from a different operator (the rest being operator self-feedback). Even with six operators available, only three serial FM modulations were possible.

XFM2 extends this concept for vastly expanded flexibility: any operator can receive modulation (and feedback) from any other operator, or any combination of these. It is possible, as an example, to have operator 1 receiving modulation from all the others, and from itself, at the same time.

Also, XFM2 features a stereo voice mixing bus. In addition to its carrier/modulator role in the modulation chain, any operator can be routed to the output, in any defined position in the stereo field, by setting the individual level sent to each channel.

Communicating with XFM2

As in any synthesizer, the sound creation in XFM2 is regulated by a set of *parameters*, which are used to specify the intended sound characteristics, nuances, responses, and expressive behavior.

There are more than 400 parameters in XFM2 to define any sound, including both the parameters for the synthesizer engine and the complete effect processors chain. A set of all parameters is then called a *program*, and each unit can run its own program.

Up to 128 programs can be saved and loaded to/from the internal memory (EEPROM). Programs can also be automatically loaded from standard MIDI Program Change messages.

As a module, XFM2 does not feature a user interface for direct interaction with its programs and parameters. It was designed to act as the sound generation core of a greater system, in which the main user performance gestures come from any standard MIDI controller.

One of the core design concepts of XFM2 was to create a sound engine module which could be controlled externally by any user interface builders would imagine, from a few LEDs and knobs to a fully-fledged touch panel or advanced interface.

There are different options to create, edit, save, and load sound programs, and to tweak all parameters in real-time:

- Using a computer connected to the USB port (see "*Controlling XFM2 via USB"*).
  This is the most immediate way to change parameters and create, save, and load programs in XFM2. A basic example of how to interact in this way can be shown in the Excel spreadsheet included in the zip file, in which simply entering the parameter values will update the device instantaneously. Using the built-in USB-to-RS232 converter in XFM2, it's simple to create a complete user interface.

- Using an external interface, using standard serial RS-232 communication.
  There are dedicated pins in the XFM2 hardware, which expose its hardware serial interface. It is a simple task to build and use any inexpensive microcontroller (Arduino, etc.) to serve as a real-time interface with physical controls for XFM2.

- Using MIDI System Exclusive messages.
  Each parameter in XFM2 can be changed by defined MIDI System Exclusive messages, sent from a MIDI controller or sequencer (see *"MIDI System Exclusive implementation chart"* for details).

For all three options, every parameter message is composed of a *parameter number*, and a *parameter value*.

Along this manual, each feature description will indicate the valid value-range for each parameter. Unless otherwise indicated, all continuous parameters have a one-byte, 256-value range from 0 to 255.

Operator structure and parameters

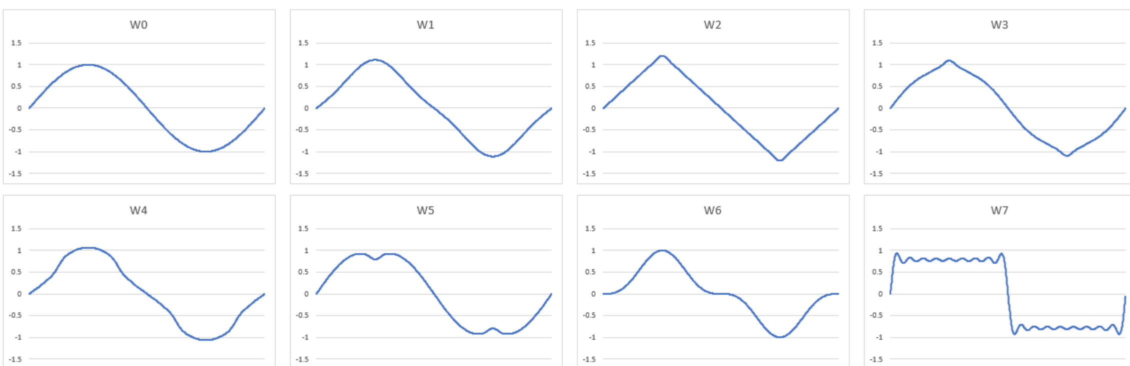
Oscillators

Each operator in XFM2 is composed of two identical and independent oscillators, capable of producing a variety of waveforms.

WAVE 1, WAVE 2	Select the waveform for Oscillator 1 and Oscillator 2, from 0 to 7. See *Oscillator Waveforms* image below for graphical waveform representations.
OSC MODE	0 = Only Oscillator 1 is active 1 = Both Oscillator 1 and 2 are active
OSC RATIO	Selects the frequency ratio between Oscillators 1 and 2.

Oscillator Waveforms

Each oscillator in any of the six operators in XFM2 can use any of the eight factory waveforms, indicated in the graphic below.
All waveforms have been carefully hand-picked with a suitable harmonic content for FM synthesis.



Interconnecting operators

As described previously, all six operators can be combined in flexible ways.

When generating sound, operators are calculated sequentially, in reverse order: first operator to be calculated is operator 6, then 5, and so on down to operator 1. Each operator has a bitwise matrix parameter which defines which operators will modulate it.

If the operators set to modulate any other operator have a greater index number (i.e. OP1 being modulated by OP6), the carrier operator receives standard FM modulation. If it has a smaller number (i.e. OP4 modulated by OP1), it receives the modulation from a feedback loop.

This approach requires a slightly more complex calculation process to define the algorithm, but in contrast, it allows for millions of radically different algorithms.

<table>
<tr><td>ALGO</td><td>Bitwise parameter, selects which operator(s) will modulate the selected operator:

<table><tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td></tr>
<tr><td>-</td><td>OP6</td><td>OP5</td><td>OP4</td><td>OP3</td><td>OP2</td><td>OP1</td><td>OUT</td></tr></table>

Bit 0 (the least significant bit) indicates if the operator output is routed to the main stereo output.</td></tr>
<tr><td>FEEDBACK</td><td>Sets the operator self-feedback level.</td></tr>
</table>

Oscillators synchronization

Oscillators in XFM2 can have free-running phase (meaning that their phase isn't affected by the notes played in any way) or be synchronized (they reset their phase with every note-on event, to a specified value).

When in synchronized mode, the initial phase of each operator can be adjusted in 90-degree intervals, useful to avoid cancellation effects. Both oscillators share the same initial phase setting.

<table>
<tr><td>SYNC</td><td>Bitwise parameter, sets individual operators to each free-run (0) or sync to note-on (1).

<table><tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td></tr>
<tr><td>-</td><td>-</td><td>OP6</td><td>OP5</td><td>OP4</td><td>OP3</td><td>OP2</td><td>OP1</td></tr></table>

Example: Use value 7 (0000 0111) to set operators 1, 2 and 3 to synchronize their phase to note-on events, and the rest to free run.</td></tr>
<tr><td>PHASE</td><td>Sets the initial phase in degrees, for the operator, as follows:
0 = 0 deg.
1 = 90
2 = 180
3 = 270</td></tr>
</table>

Pitch controls

Each operator can either have a pitch that tracks the incoming MIDI controller notes or generate a definable fixed pitch.

<table>
<tr><td>MODE</td><td>Bitwise parameter, defines if each operator either tracks the keyboard (0) or produces a fixed tone (1).

<table><tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td></tr>
<tr><td>-</td><td>-</td><td>OP6</td><td>OP5</td><td>OP4</td><td>OP3</td><td>OP2</td><td>OP1</td></tr></table>

Example: Value 4 (0000 0100) will set operator 3 to produce a fixed-pitch tone, and all the rest to track the incoming MIDI notes.</td></tr>
<tr><td>RATIO</td><td>Sets the pitch ratio against the standard MIDI note-based pitch that the operator produces, in integer values (i.e. 3 = 3:1 ratio).</td></tr>
<tr><td>RATIO FINE</td><td>Sets the pitch ratio against the standard MIDI note-based pitch that the operator produces, in fractions of an octave (0 = 1:1 ratio, 255 ~= 2:1 ratio).</td></tr>
<tr><td>FINE</td><td>Adjusts the operator pitch, in cents (128 = 0 cents).</td></tr>
</table>

Operator level

Each operator in XFM2 has an output level adjustment, and individual output levels for the left and right stereo channels. This allows to place each operator in any position in the stereo field.

LEVEL	Sets the operator output level. NOTE: output level is logarithmic, so greater values have more perceived effect than smaller values.
LEVEL L, LEVEL R	Define the levels in which the operator contributes to each channel in the stereo output mix.
VELO SENS	Defines how much the operator level will change with incoming MIDI velocity values. NOTE: this control is logarithmic, so greater values have a greater effect. This control is also affected by the global VELOCITY OFFSET parameter.

Keyboard Tracking

In FM synthesis, is very important to control the operator's modulation depth as sounds change in pitch upward or downward. Programs that offer the best expressive behavior across all the keyboard have a very careful adjustment of the keyboard tracking.

KEY BP	Sets the keyboard breakpoint. This means, the note number that splits the left and right keyboard tracking curves and depth.
KEY LDEPTH	Adjusts the depth of the scaling at the left of the breakpoint.
KEY RDEPTH	Adjusts the depth of the scaling at the right of the breakpoint.
KEY LCURVE	Sets the curve at the left of the breakpoint. 0 = exp - 1 = lin - 2 = lin + 3 = exp +
KEY RCURVE	Sets the curve at the right of the breakpoint. 0 = exp - 1 = lin - 2 = lin + 3 = exp +

Amplitude Envelope Generators

Each operator in XFM2 has its own *amplitude envelope generator*, which controls its output level as a function of the time.

All envelope generators are seven-segment *DAD1D2SR1R2*: Delay, Attack, Decay-1, Decay-2, Sustain, Release-1, Release-2.

Each envelope segment has separately adjustable parameters for *time* and *level* (except for *delay*, which has no level adjustment). Envelopes can be looped, with two different looping return points. Envelope times can also be set to track the keyboard (so envelopes are shorter or longer with higher notes), and each envelope generator can have its own keyboard tracking setting.

The envelope generators in XFM2 are extremely fast, as they are all fully recalculated on every sample, over 60,000 times every second.

When the XFM2 module is maxed in polyphony, there are 64 voices running 7 envelope generators each (one amplitude EG for each operator, plus one voice pitch EG), for a total of 448 simultaneous samplerate-grade envelope generators, each one with 14 user-adjustable parameters (!).

The ability to do this reliably while being battery-efficient is one of the most evident advantages of having an advanced FPGA device to do the job.

<table>
<tr><td>DLY</td><td>Delay time before the EG starts.</td></tr>
<tr><td>L0, L1, L2, L3, L4, L5</td><td>Segment final levels for Attack, Decay-1, Decay-2, Sustain, Release-1 and Release-2 stages, respectively.</td></tr>
<tr><td>R0, R1, R2, R3, R4, R5</td><td>Segment times for Attack, Decay-1, Decay-2, Sustain, Release-1 and Release-2 stages, respectively.</td></tr>
<tr><td>RATE KEY</td><td>Sets how the EG times change with the generating MIDI notes.</td></tr>
<tr><td>EG LOOP</td><td>Bitwise parameter, sets each envelope generator to normal mode, or loop.
MSB LSB<table><tr><td>-</td><td>PITCH</td><td>OP6</td><td>OP5</td><td>OP4</td><td>OP3</td><td>OP2</td><td>OP1</td></tr></table>Example: Value 9 (0000 1001) sets the envelope generators of operators 1 and 4 to loop, and the rest to normal.</td></tr>
<tr><td>EG LOOP SEG</td><td>When set to loop, envelope generators will loop when they reach the sustain stage. This bitwise parameter defines which stage the EG loops to: if 0, it’ll loop to the Attack segment, if 1 it’ll loop to Decay-1 segment.
MSB LSB<table><tr><td>-</td><td>PITCH</td><td>OP6</td><td>OP5</td><td>OP4</td><td>OP3</td><td>OP2</td><td>OP1</td></tr></table>Example: Value 9 (0000 1001) sets the envelope generators of operators 1 and 4 to restart from Attack segment on each loop iteration.</td></tr>
</table>

Amplitude Modulation Sensitivity

Any operator can receive amplitude (output level) modulation from the LFO and MIDI controls.
The sensitivity to this modulation can be adjusted individually for each operator.

AMS	Sets the amplitude modulation sensitivity for this operator.

Pitch Modulation Sensitivity

Any operator can also receive pitch modulation from the LFO and MIDI controls. The sensitivity to this modulation can be adjusted individually for each operator.

PMS	Sets the pitch modulation sensitivity for this operator.

Voice structure and parameters

Voice basics

Besides the set of parameters that can be adjusted individually for any of the six operators, each voice in XFM2 have a set of voice parameters, that apply to all operators equally.

Pitch Envelope Generator

The Pitch EG is used to adjust the pitch of all operators in each voice simultaneously as a function of the time.

It has a similar structure to the operator Amplitude EGs, except that it is bipolar (center value is 128) so any segment can change pitch in any direction with respect to the default pitch.

<table>
<tr><td>DLY</td><td>Delay time before the EG starts.</td></tr>
<tr><td>L0, L1, L2, L3, L4, L5</td><td>Segment final levels for Attack, Decay-1, Decay-2, Sustain, Release-1 and Release-2 stages, respectively.
Center value is 128.</td></tr>
<tr><td>R0, R1, R2, R3, R4, R5</td><td>Segment times for Attack, Decay-1, Decay-2, Sustain, Release-1 and Release-2 stages, respectively.</td></tr>
<tr><td>RATE KEY</td><td>Sets how the EG times change with the generating MIDI notes.</td></tr>
<tr><td>EG LOOP</td><td>Bitwise parameter, sets each envelope generator to normal mode, or loop.

MSB … LSB
| - | - | OP6 | OP5 | OP4 | OP3 | OP2 | OP1 |

Example: Value 9 (0000 1001) sets the envelope generators of operators 1 and 4 to loop, and the rest to normal.</td></tr>
<tr><td>EG LOOP SEG</td><td>When set to loop, envelope generators will loop when they reach the sustain stage. This bitwise parameter defines which stage the EG loops to: if 0, it’ll loop to the Attack segment, if 1 it’ll loop to Decay-1 segment.

MSB … LSB
| - | - | OP6 | OP5 | OP4 | OP3 | OP2 | OP1 |

Example: Value 9 (0000 1001) sets the envelope generators of operators 1 and 4 to restart from Attack segment on each loop iteration.</td></tr>
<tr><td>RANGE</td><td>Defines the Pitch EG range, in octaves.</td></tr>
<tr><td>VELO</td><td>Sets how much the Pitch EG range changes with incoming MIDI velocity, in octaves.</td></tr>
</table>

Low Frequency Oscillator

The Low Frequency Oscillator (LFO) in XFM2 is used as a modulation source to dynamically alter a sound over time.
It can be applied to any operator pitch (see PMS), amplitude (see AMS), either directly or using the MIDI controllers for Wheel, Aftertouch, Breath or Foot controllers as performance gestures.

LFOs in XFM2 are polyphonic, so each voice has its own (this means that when maxed in polyphony, there are 64 LFOs running simultaneously).

WAVE	Sets the LFO waveform: 0 = Triangle 1 = Square 2 = Saw up 3 = Saw down 4 = Sine 5 = Random
SPEED	Sets the LFO speed.
SYNC	Defines the LFO behavior. 0 = Single, Free running 1 = Single, Key sync 2 = Multi, Free running 3 = Multi, Key sync In the *single* modes, all voices LFOs are in phase, even when key sync'ed (a note-on message will restart all LFOs). IN the *multi* modes, LFOs in all voices are completely individual.
FADE	Sets the time for the LFO effect to ramp up to the maximum.
DEPTH PITCH	Sets the fixed LFO effect depth applied to pitch (individual operators can receive different values, see PMS).
DEPTH AMP	Sets the fixed LFO effect depth applied to amplitude (individual operators can receive different values, see AMS).

Program settings

The Program settings are parameters that affect all voices simultaneously.

BEND UP	XFM2 allows for individual adjustments of the Pitch Bend MIDI control, for up and down. Sets the pitch range to use when upward MIDI Pitch Bend messages are received.
BEND DOWN	Sets the pitch range to use when downward MIDI Pitch Bend messages are received.
TRANSPOSE	Sets the overall transpose, in semitones. Center value is 24 (no transpose). Lower values will transpose down, higher values will transpose up.
VOLUME	Set the overall synthesizer voice mix volume.
PAN	Set the overall synthesizer voice mix panoramic position.
VELOCITY OFFSET	Defines a value to be added to incoming MIDI velocity values, as to compensate for different keyboards and controllers' physical response.
LEGATO MODE	Sets the synthesizer unit into Polyphonic (0) or Monophonic/Legato (1) operation.
PORTA MODE	XFM2 feature *portamento*, a pitch-sliding effect where the pitch of each new note glides from the previous one at an adjustable time. This parameter sets the portamento mode: 0 = Off 1 = Always (any new note will glide in from previous one pitch value) 2 = Fingered (only legato notes will glide in) Portamento in XFM2 is polyphonic or monophonic according to the LEGATO MODE parameter value.
PORTA TIME	Sets the time for the Portamento effect.

Modulations Section

Pitch

The modulation Pitch parameters adjust how MIDI controls modify the pitch of each note.

AFTER	Sets how the pitch changes with MIDI Aftertouch messages.
BREATH	Sets how the pitch changes with MIDI Breath Controller messages (CC 02).
FOOT	Sets how the pitch changes with MIDI Foot Controller messages (CC 04).
RND	Sets how much each note pitch changes randomly (modulated from an internal random-value generator).

Pitch LFO

The Pitch LFO parameters adjust how MIDI controls modify the application of LFOs to operator pitches.

AFTER	Regulates how much of the LFO is applied to pitch with MIDI Aftertouch messages.
WHEEL	Sets how much of the LFO is applied to pitch with MIDI Modulation Wheel messages (CC 01).
BREATH	Sets how much of the LFO is applied to pitch with MIDI Breath Controller messages (CC 02).
FOOT	Sets how much of the LFO is applied to pitch with MIDI Foot Controller messages (CC 04).

Amplitude LFO

The Amplitude LFO parameters adjust how MII controls modify the application of LFOs to operator amplitudes.

AFTER	Regulates how much of the LFO is applied to amplitude with MIDI Aftertouch messages.
WHEEL	Sets how much of the LFO is applied to amplitude with MIDI Modulation Wheel messages (CC 01).
BREATH	Sets how much of the LFO is applied to amplitude with MIDI Breath Controller messages (CC 02).
FOOT	Sets how much of the LFO is applied to amplitude with MIDI Foot Controller messages (CC 04).

EG Bias

The EG Bias parameters adjust how MIDI controls adjust the offset position for the amplitude EGs.

AFTER	Regulates the amplitude offset with MIDI Aftertouch messages.
WHEEL	Sets the amplitude offset with MIDI Modulation Wheel messages (CC 01).
BREATH	Sets the amplitude offset with MIDI Breath Controller messages (CC 02).
FOOT	Sets the amplitude offset with MIDI Foot Controller messages (CC 04).

Arpeggiator

Each synthesizer unit in XFM2 features a simple, yet useful arpeggiator.

ARP MODE	Selects the arpeggiator mode, as follows: 0 = Off 1 = Up 2 = Down 3 = Up/Down 4 = As played 5 = Random
TEMPO	Sets the arpeggiator tempo in bpm, 50~255. If set to 0, the arpeggiator will follow the incoming MIDI Clock tempo.
MUL	Defines a tempo multiplicator. This allows to create subdivisions, and rhythmic variations on the main tempo.
OCTAVES	Sets the number of octaves the arpeggiator will repeat the pattern in.

Alternate tunings

XFM2 features up to 256 alternate tunings and keyboard definition maps.
Each program can have its own alternate tuning from the global selection. The selected tuning is recalled with the program, so both units could have different tunings.

TUNING	Selects the alternate tuning. See *Factory Tunings* for more information on available tuning definitions.

Creating user alternate tunings

All the factory alternate tuning definitions were created by converting standard *Scala* (.scl) tuning definition files.
Scala is a powerful and free software tool for experimentation with musical tunings, such as just intonation scales, equal and historical temperaments, microtonal and macrotonal scales, and non-Western scales.

The zip file with XFM2 binaries includes the utility to convert Scala files into the XFM2 binary format, and instructions on how to create a different factory tuning definition file and uploading those into XFM2.

Effects Processors

Each unit in XFM2 includes a powerful seven-stage effects processor, featuring lo-fi effects (bitcrusher, decimator), low-pass and high-pass filters, modulation effects (chorus/flanger, phaser, amplitude modulation) and delay.

All seven effects in both units (14 in total) can be running simultaneously. Additionally, there are two choices for inter-effects routing, which allow different effect chain combinations.

Bitcrusher

The Bitcrusher effect reduces the bit depth of the incoming signal. Ideal to create chiptune-like sounds, or 8-bit sounds.

DEPTH	Sets the number of bits to reduce (1-24). WARNING: high values can result in loud sounds.

Decimator

The Decimator effect reduces the samplerate of the incoming signal, to create grainy, gritty and metallic tones.

DEPTH	Adjusts the amount of samplerate reduction.

Filter

The Filter effect includes two one-pole filters: one low-pass and one high-pass, in series. The filters are ideal to tone down the effect of the two previous effects, or a heavily modulated program's brilliance.

LO	Sets the low-pass filter cutoff.
HI	Sets the high-pass filter cutoff.

Chorus/Flanger

The stereo Chorus/Flanger effect in XFM2 is capable of a broad variety of modulation effects, from very subtle to dramatic.

DRY	Adjusts the level of the dry signal (no effect).
WET	Adjust the level of the wet signal (full effect).
MODE	Adjusts the effect mode, as follows: 0 = Chorus (long) 1 = Chorus (short) 2 = Flanger (long) 3 = Flanger (short)
SPEED	Sets the modulation speed.
DEPTH	Sets the modulation depth.
FEEDBACK	Adjusts the feedback level for the effect (output signal resent to the input).
LR PHASE	Sets the L-R phase (128 for quadrature phase).

Phaser

The stereo Phaser effect is a modulated multi-stage all-pass filter design, with 4 to 12 stages.

DRY	Adjusts the level of the dry signal (no effect).
WET	Adjust the level of the wet signal (full effect).
MODE	Sets the phaser mode, as follows: 0 = Mono 1 = Stereo 2 = Cross (each channel's feedback into the other)
SPEED	Sets the modulation speed.
DEPTH	Sets the modulation depth.
OFFSET	Sets the center frequency point for the effect sweep.
STAGES	Sets the number of all-pass stages, from 4 to 12.
FEEDBACK	Adjusts the feedback level for the effect (output signal resent to the input).
LR PHASE	Sets the L-R phase (128 for quadrature phase).

Amplitude Modulation

The Amplitude Modulation effect modulates the input signal in amplitude with an internally generated sine, to create different effects ranging from tremolo, to auto-panning, to high-frequency amplitude modulation.

DEPTH	Sets the modulation depth.
SPEED	Sets the modulation speed.
RANGE	This parameter adjusts the speed range. Can be thought as a coarse speed control.
LR PHASE	Sets the L-R phase (128 for quadrature phase).

Delay

The Delay effect produces all kinds of delay-based effects, such as doubling, slapback echo, mono/stereo echo or ping delay. The effect feedback loop includes both low-pass and high-pass filters.

DRY	Adjusts the level of the dry signal (no effect).
WET	Adjust the level of the wet signal (full effect).
MODE	Sets the delay mode, as follows: 0 = Stereo 1 = Cross (each channel's feedback into the other) 2 = Bounce (echoes bounce in the stereo field)
TIME	Adjusts the delay time. When set to 0, the delay follows the TEMPO setting.
FEEDBACK	Adjusts the feedback level for the effect (output signal resent to the input).
LO	Regulates the low-pass filter frequency response for the feedback loop.
HI	Regulates the high-pass filter frequency response for the feedback loop.
TEMPO	Adjusts the delay tempo in bpm, 50~255 (operational only when the TIME parameter is set to 0).
MUL	Sets a multiplier for the delay tempo.
DIV	Sets a divider for the delay tempo. Combining this value with the multiplier, complex rhythmic effects can be achieved.

Effects routing

Two different routing effects are selectable in each effect processor unit.

FX ROUTING	Selects the effects routing, as follows: 0 = Bitcrusher → Decimator → Filter → Chorus → Phaser → AM → Delay 1 = Bitcrusher → Decimator → Filter → Delay → Chorus → Phaser → AM

Global Reverb

Both synthesizer-effect units feed a common global reverb. Each unit can contribute to the dry and wet reverb mixes individually.

DRY	Adjusts the level of the dry signal (no effect). NOTE: per-unit parameter.
WET	Adjust the level of the wet signal (full effect). NOTE: per-unit parameter.
MODE	Sets the reverb mode, as follows: 0 = Plate 1 = Hall NOTE: settings are recalled by unit #0 setting.
DECAY	Adjusts the reverb decay time. NOTE: settings are recalled by unit #0 setting.
DAMP	Sets the reverb high-frequency damping. NOTE: settings are recalled by unit #0 setting.

Output level

The output level parameter serves as a final adjustment for each unit, after all effects are processed.

OUTPUT	Adjusts the program output level.

Performance Controls

Each program in XFM2 can hold up to four Performance Controls. Each Performance Control can be mapped to any program parameter, for real-time control.

This means, for instance, that Program 1 could have LFO Speed, OP1 Ratio, OP3 Level and OP4 Attack Time assigned to react to the Performance Control MIDI messages (CC 70 to CC 73), while Program 2 could have a completely different set of parameters mapped.

Each Performance Control is specified as two 7-bit numbers.

`CTL1H`, `CTL1L`	Set Performance Control 1 7-bit MSB and LSB, respectively (see *Calculating 7-bit numbers* for details). Performance Control 1 responds to MIDI Continuous Control 70. Example: `CTL1H` = 0, `CTL1H`= 180 for OP1 Feedback.
`CTL2H`, `CTL2L`	Set Performance Control 2 7-bit MSB and LSB, respectively (see *Calculating 7-bit numbers* for details). Performance Control 2 responds to MIDI Continuous Control 71. Example: `CTL2H` = 2, `CTL2H`= 48 for Delay Feedback.
`CTL3H`, `CTL3L`	Set Performance Control 2 7-bit MSB and LSB, respectively (see *Calculating 7-bit numbers* for details). Performance Control 3 responds to MIDI Continuous Control 72. Example: `CTL3H` = 1, `CTL3H`= 11 for Pitch EG Velocity Range.
`CTL4H`, `CTL4L`	Set Performance Control 2 7-bit MSB and LSB, respectively (see *Calculating 7-bit numbers* for details). Performance Control 1 responds to MIDI Continuous Control 73. Example: `CTL4H` = 1, `CTL4H`= 23 for LFO Speed.

Global settings

MIDI Channel

Each unit in XFM2 can be set to its own, independent MIDI Channel.
This is a global setting and persists on power-down. Please refer to *“Controlling XFM2 via USB2”* for details on how to change MIDI channels.

Layer Mode

It is possible to set both units to the same MIDI channel, to have both parts reacting to the same MIDI notes. This is, having both units as layers.

However, when doing this, both units will select the same program when MIDI Program Change messages are sent to XFM2, resulting in two identical sounds being played.

To avoid this, there’s a global setting to turn XFM2 into “Layer Mode”. In this mode, both units will play together, reacting to any MIDI note in any channel. However, each unit will only react to parameter and program changes in its selected MIDI channel.

This allows to play two selected programs layered.

Controlling XFM2 via USB

XFM2 features a built-in USB-to-serial converter. This means that when XFM2 is connected to a computer using an USB cable, a serial communication port (COM) can be used to send and receive commands and data to it. The default communication port is COM7, but it can be changed to be any port from your operating system. The virtual port in XFM2 is very fast, and can be open as 500,000 bps, 8-bits, 1-stop bit, no parity.

You can use any software terminal program to send and receive serial commands to XFM2. The following table describe the supported commands.

Command	Value 1 (dec)	Value 2 (dec)	Value 3 (dec)	Result (dec)	Function
d	-	-	-	512 bytes	Displays all parameter values for active program.
i	-	-	-	0	Initializes active program.
r	Program number (0-127)	-	-	0	Loads a program from EEPROM.
w	Program number (0-127)	-	-	0	Writes active program into EEPROM.
s	Parameter number (0-254, or 255 if parameter is greater than 254).	Parameter number minus 255. Only required if the parameter number is greater than 254.	Parameter value	-	Sets specified parameter to specified value.
g	Parameter number (0-254, or 255 if parameter is greater than 254).	Parameter number minus 255. Only required if the parameter number is greater than 254.	-	Parameter value.	Gets active value for specified parameter.
1	-	-	-	0	Activates first unit (all subsequent communication will be with synth/fx unit #0).
2	-	-	-	1	Activates second unit (all subsequent communication will be with synth/fx unit #1).
$	-	-	-	-	Initializes EEPROM. WARNING: all programs will be lost.
*	10	MIDI Channel, 0 (omni) or 1-16.	-	MIDI Channel	Sets unit #1 MIDI channel.
*	11	MIDI Channel, 0 (omni) or 1-16.	-	MIDI Channel	Sets unit #2 MIDI channel.
*	12	0 or 1	-	0	Sets LAYER mode off (0) or on (1).

Appendices

Factory tunings

XFM2 can use up to 256 different tuning structures, or scales. Additionally, tuning banks can be uploaded into the unit by using the *XLoad* utility (Windows).
Scales are not limited to 12-tones per octave; any MIDI note can be tuned to any desired pitch.

The factory tuning bank consist of a selection of tunings created with the *Scala* software tool. *Scala* is a powerful and free software tool for experimentation with musical tunings, such as just intonation scales, equal and historical temperaments, microtonal and macrotonal scales, and non-Western scales.

XFM2 includes the *WScala* converter utility (Windows), which allows the conversion of any Scala (.scl) file into a binary which can be added into an XFM bank.

This is the list of the tunings in the factory bank:

#	Scala file name	Scala file description
0	00 - Standard.scl	Standard Tuning
1	05-19.scl	5 out of 19-tET
2	05-22.scl	Pentatonic "generator" of 09-22.scl
3	05-24.scl	5 out of 24-tET, symmetrical
4	06-41.scl	Hexatonic scale in 41-tET, Magic-6
5	07-19.scl	Nineteen-tone equal major
6	07-31.scl	Strange diatonic-like strictly proper scale
7	07-37.scl	Miller's Porcupine-7
8	08-11.scl	8 out of 11-tET
9	08-13.scl	8 out of 13-tET
10	08-19.scl	8 out of 19-tET, Mandelbaum
11	08-37.scl	Miller's Porcupine-8
12	09-15.scl	Charyan scale of Andal, Boudewijn Rempt (1999), 1/1=A
13	09-19.scl	9 out of 19-tET, Mandelbaum. Negri[9]
14	09-19a.scl	Second strictly proper 9 out of 19 scale
15	09-22.scl	Trivalent scale in 22-tET, TL 05-12-2000
16	09-23.scl	9 out of 23-tET, Dan Stearns
17	09-29.scl	Cycle of g=124.138 in 29-tET (Negri temperament)
18	09-31.scl	Scott Thompson scale 724541125
19	10-13-58.scl	Single chain pseudo-MOS of major and neutral thirds in 58-tET
20	10-13.scl	10 out of 13-tET MOS, Carl Lumma, TL 21-12-1999
21	10-19.scl	10 out of 19-tET, Mandelbaum. Negri[10]
22	10-29.scl	10 out of 29-tET, chain of 124.138 cents intervals, Keenan
23	11-18.scl	11 out of 18-tET, g=333.33, TL 27-09-2009
24	11-19-gould.scl	11 out of 19-tET, Mark Gould (2002)
25	11-19-krantz.scl	11 out of 19-tET, Richard Krantz
26	11-19-mclaren.scl	11 out of 19-tET, Brian McLaren. Asc: 311313313 Desc: 313131313
27	11-23.scl	11 out of 23-tET, Dan Stearns
28	11-31.scl	Jon Wild, 11 out of 31-tET, g=7/6, TL 9-9-1999
29	11-34.scl	Erv Wilson, 11 out of 34-tET, chain of minor thirds

30	11-37.scl	Jake Freivald, 11 out of 37-tET, g=11/8, TL 22-08-2012
31	11-limit-only.scl	11-limit-only
32	12-17.scl	12 out of 17-tET, chain of fifths
33	12-22.scl	12 out of 22-tET, chain of fifths
34	12-22h.scl	Hexachordal 12-tone scale in 22-tET
35	12-27.scl	12 out of 27, Herman Miller's Galticeran scale
36	12-31.scl	12 out of 31-tET, meantone Eb-G#
37	12-31_11.scl	11-limit 12 out of 31-tET, George Secor
38	12-43.scl	12 out of 43-tET (1/5-comma meantone)
39	12-46.scl	12 out of 46-tET, diaschismic
40	12-46p.scl	686/675 comma pump scale in 46-tET
41	12-50.scl	12 out of 50-tET, meantone Eb-G#
42	12-79mos159et.scl	12-tones out of 79 MOS 159ET, Splendid Beat Rates Based on Simple Frequencies version, C=262hz
43	12-yarman24a.scl	12-tones out of Yarman24a, circulating in the style of Rameau's Modified Meantone Temperament
44	12-yarman24b.scl	12-tones out of Yarman24b, circulating in the style of Rameau's Modified Meantone Temperament
45	12-yarman24c.scl	12-tones out of Yarman24c, circulating in the style of Rameau's Modified Meantone Temperament
46	12-yarman24d.scl	12-tones out of Yarman24d, circulating in the style of Rameau's Modified Meantone Temperament
47	13-19.scl	13 out of 19-tET, Mandelbaum
48	13-22.scl	13 out of 22-tET, generator = 5
49	13-30t.scl	Tritave with 13/10 generator, 91/90 tempered out
50	13-31.scl	13 out of 31-tET Hemiwürschmidt[13]
51	14-19.scl	14 out of 19-tET, Mandelbaum
52	14-26.scl	Two interlaced diatonic in 26-tET, tetrachordal. Paul Erlich (1996)
53	14-26a.scl	Two interlaced diatonic in 26-tET, maximally even. Paul Erlich (1996)
54	15-37.scl	Miller's Porcupine-15
55	15-46.scl	15
56	16-139.scl	g=9 steps of 139-tET. Gene Ward Smith "Quartaminorthirds" 7-limit temperament
57	16-145.scl	Magic[16] in 145-tET
58	16-31.scl	Armodue semi-equalizzato
59	17-31.scl	17 out of 31, with split C#/Db, D#/Eb, F#/Gb, G#/Ab and A#/Bb
60	17-53.scl	17 out of 53-tET, Arabic Pythagorean scale, Safiyuddîn Al-Urmawî (Safi al-Din)
61	19-31.scl	19 out of 31-tET, meantone Gb-B#
62	19-31ji.scl	A septimal interpretation of 19 out of 31 tones, after Wilson, XH7+8
63	19-36.scl	19 out of 36-tET, Tomasz Liese, Tuning List, 1997
64	19-50.scl	19 out of 50-tET, meantone Gb-B#
65	19-53.scl	19 out of 53-tET, Larry H. Hanson (1978), key 8 is Mason Green's 1953 scale
66	19-55.scl	19 out of 55-tET, meantone Gb-B#
67	19-any.scl	Two out of 1/7 1/5 1/3 1 3 5 7 CPS
68	20-31.scl	20 out of 31-tET

69	20-55.scl	20 out of 55-tET, J. Chesnut: Mozart's teaching of intonation, JAMS 30/2 (1977)
70	21-any.scl	2)7 1.3.5.7.9.11.13 21-any, 1.3 tonic
71	22-100.scl	MODMOS with 10 and 12-note chains of fifths by Gene Ward Smith, similar to Pajara
72	22-100a.scl	Alternative version with 600 cents period
73	22-41.scl	22 out of 41 by Stephen Soderberg, TL 17-11-98
74	22-46.scl	22 shrutis out of 46-tET by Graham Breed
75	22-53.scl	22 shrutis out of 53-tET
76	24-41.scl	24 out of 41-tET, g=neutral third, 22 neutral triads, Op de Coul (2001), Hemififths-24
77	24-60.scl	12 and 15-tET mixed. Novaro (1951)
78	24-80.scl	Regular 705-cent temperament, 24 of 80-tET
79	24-94.scl	24 tone schismic temperament in 94-tET, Gene Ward Smith (2002)
80	28-any.scl	6)8 1.3.5.7.9.11.13.15 28-any, only 26 tones
81	30-29-min3.scl	30/29 x 29/28 x 28/27 plus 6/5
82	31-171.scl	Tertiaseptal-31 in 171-tET, g=11\171
83	46_72.scl	46 note subset of 72-tET containing the 17-limit otonalities and utonalities by Rick Tagawa
84	53-commas.scl	so-called 1/9 comma division of Turkish Music by equal division of 9/8 into 9 equal string lengths
85	56-any.scl	3)8 1.3.5.7.9.11.13.15 56-any, 1.3.5 tonic, only 48 notes
86	67-135.scl	67 out of 135-tET by Ozan Yarman, g=17.7777
87	70-any.scl	4)8 1.3.5.7.11.13.17.19 70-any, tonic 1.3.5.7
88	79-159.scl	79 out of 159-tET MOS by Ozan Yarman, 79-tone Tuning & Theory For Turkish Maqam Music
89	79-159beats.scl	79 MOS 159tET Splendid Beat Rates Based on Simple Frequencies, C=262 hz
90	79-159first.scl	79 MOS 159-tET original pure fourths version
91	79-159ji.scl	79 MOS 159-tET Just Intonation Ratios
92	79-159_arel-ezgi-uzdilek.scl	Arel-Ezgi-Uzdilek style of 11 fifths up, 12 down from tone of origin in 79 MOS 159-tET
93	79-159_equidistant5ths.scl	79 MOS 159-tET equi-distant fifths from pure 3:2 version.
94	79-159_splendidbeating.scl	79 MOS 159-tET Splendid Beat Rates Based on Simple Frequencies, C=262 hz
95	80-159.scl	80 out of 159-tET MOS by Ozan Yarman, 79-tone Tuning & Theory For Turkish Maqam Music
96	80-159beats.scl	80 MOS 159tET Splendid Beat Rates Based on Simple Frequencies, C=262 hz
97	80-159_splendidbeating.scl	80 MOS 159-tET Splendid Beat Rates Based on Simple Frequencies, C=262 hz
98	abell1.scl	Ross Abell's French Baroque Meantone 1, a'=520 Hz
99	abell2.scl	Ross Abell's French Baroque Meantone 2, a'=520 Hz
100	abell3.scl	Ross Abell's French Baroque Meantone 3, a'=520 Hz
101	abell4.scl	Ross Abell's French Baroque Meantone 4, a'=520 Hz
102	abell5.scl	Ross Abell's French Baroque Meantone 5, a'=520 Hz
103	abell6.scl	Ross Abell's French Baroque Meantone 6, a'=520 Hz

104	abell7.scl	Ross Abell's French Baroque Meantone 7, a'=520 Hz
105	abell8.scl	Ross Abell's French Baroque Meantone 8, a'=520 Hz
106	abell9.scl	Ross Abell's French Baroque Meantone 9, a'=520 Hz
107	ad-dik.scl	Amin Ad-Dik, 24-tone Egyptian tuning, d'Erlanger vol.5, p. 42
108	aeolic.scl	Ancient Greek Aeolic, also tritriadic scale of the 54:64:81 triad
109	aeu-41 ratios.scl	AEU extended to quasi-cyclic 41-tones in simple ratios
110	aeu-41.scl	AEU extended to 41-quasi equal tones by Ozan Yarman
111	agricola.scl	Agricola's Monochord, Rudimenta musices (1539)
112	agricola_p.scl	Agricola's Pythagorean-type Monochord, Musica instrumentalis deudsch (1545)
113	akea46_13.scl	46
114	al-din.scl	Safi al-Din's complete lute tuning on 5 strings 4/3 apart
115	blackjack.scl	21 note MOS of "MIRACLE" temperament, Erlich & Keenan, miracle1.scl,TL 2-5-2001
116	blackjackg.scl	Blackjack on G-D
117	blackjack_r.scl	Rational "Wilson/Grady"-style version, Paul Erlich, TL 28-11-2001
118	blackjack_r2.scl	Another rational Blackjack maximising 1:3:7:9:11, Paul Erlich, TL 5-12-2001
119	blackjack_r3.scl	7-Limit rational Blackjack, Dave Keenan, TL 5-12-2001
120	blackjb.scl	Marvel (1,1) tuning of pipedum_21b
121	blackopkeegil1.scl	Blacksmith-Opossum-Keemun-Gilead Wakalix 1
122	blackopkeegil2.scl	Blacksmith-Opossum-Keemun-Gilead Wakalix 2
123	blackwoo.scl	Irregular Blackjack from marvel woo tempering of Cartesian scale below
124	blackwood.scl	Blackwood temperament, g=84.663787, p=240, 5-limit
125	blackwood_6.scl	Easley Blackwood, whole tone scale, arrangement of 4:5:7:9:11:13, 1/1=G, p.114
126	blackwood_9.scl	Blackwood, scale with pure triads on I II III IV VI and dom.7th on V. page 83
127	blasquinten.scl	Blasquintenzirkel. 23 fifths in 2 oct. C. Sachs, Vergleichende Musikwiss. p. 28
128	blueji-cataclysmic.scl	Blueji tempered in 13-limit POTE-tuned cataclysmic
129	bluesmarvwoo.scl	Marvel woo version of Graham Breed's Blues scale
130	bluesrag.scl	Ragismic tempered bluesji in 8419-tET
131	bobrova.scl	Bobrova Cheerful 12 WT based on *19 EDL
132	bobro_phi.scl	Cameron Bobro's phi scale, TL 06-05-2009
133	bobro_phi2.scl	Cameron Bobro, first 5 golden cuts of Phi, TL 09-05-2009
134	boeth_chrom.scl	Boethius's Chromatic. The CI is 19/16
135	boeth_enh.scl	Boethius's Enharmonic, with a CI of 81/64 and added 16/9
136	bohlen-eg.scl	Bohlen-Pierce with two tones altered by minor BP diesis, slightly more equal
137	bohlen-p.scl	See Bohlen, H. 13-Tonstufen in der Duodezime, Acustica 39: 76-86 (1978)
138	bohlen-p_9.scl	Bohlen-Pierce subscale by J.R. Pierce with 3:5:7 triads
139	bohlen-p_9a.scl	Pierce's 9 of 3\13, see Mathews et al., J. Acoust. Soc. Am. 84, 1214-1222
140	bohlen-p_eb.scl	Bohlen-Pierce scale with equal beating 5/3 and 7/3
141	bohlen-p_ebt.scl	Bohlen-Pierce scale with equal beating 7/3 tenth
142	bohlen-p_ebt2.scl	Bohlen-Pierce scale with equal beating 7/5 tritone

143	bohlen-p_et.scl	13-tone equal division of 3/1. Bohlen-Pierce equal approximation
144	bohlen-p_ring.scl	Todd Harrop, symmetrical ring of Bohlen-Pierce enharmonics using 4 major and 8 minor dieses (2012)
145	bohlen-p_sup.scl	Superparticular Bohlen-Pierce scale
146	bohlen47.scl	Heinz Bohlen, mode of 4\47 (1998), www.huygens-fokker.org/bpsite/pythagorean.html
147	bohlen47r.scl	Rational version, with alt.9 64/49 and alt.38 40/13
148	bohlen5.scl	5-limit version of Bohlen-Pierce
149	bohlen_11.scl	11-tone scale by Bohlen, generated from the 1/1 3/2 5/2 triad
150	bohlen_8.scl	See Bohlen, H. 13-Tonstufen in der Duodezime, Acustica 39: 76-86 (1978)
151	bohlen_arcturus.scl	Paul Erlich, Arcturus-7, TOP tuning (15625/15309 tempered)
152	bohlen_canopus.scl	Paul Erlich, Canopus-7, TOP tuning (16875/16807 tempered)
153	bohlen_coh.scl	Differentially coherent Bohlen-Pierce, interval=2
154	bohlen_coh2.scl	Differentially coherent Bohlen-Pierce, interval=1,2, subharmonic=25
155	bohlen_coh3.scl	Differentially coherent Bohlen-Pierce, interval=1, subharmonic=75
156	bohlen_delta.scl	Bohlen's delta scale, a mode B-P, see Acustica 39: 76-86 (1978)
157	bohlen_diat_top.scl	BP Diatonic, TOP tuning (245/243 tempered)
158	bohlen_d_ji.scl	Bohlen's delta scale, just version. "Dur" form, "moll" is inversion.
159	bohlen_enh.scl	Bohlen-Pierce scale, all enharmonic tones
160	bohlen_eq.scl	Most equal selection from all enharmonic Bohlen-Pierce tones
161	bohlen_gamma.scl	Bohlen's gamma scale, a mode of the Bohlen-Pierce scale
162	bohlen_g_ji.scl	Bohlen's gamma scale, just version
163	bohlen_harm.scl	Bohlen's harmonic scale, inverse of lambda
164	bohlen_h_ji.scl	Bohlen's harmonic scale, just version
165	bohlen_lambda.scl	Bohlen's lambda scale, a mode of the Bohlen-Pierce scale
166	bohlen_lambda_pyth.scl	Dave Benson's BP-Pythagorean scale, lambda mode of bohlen_pyth.scl
167	bohlen_l_ji.scl	Bohlen's lambda scale, just version
168	bohlen_mean.scl	1/3 minor BP diesis (245/243) tempered 7/3 meantone scale
169	bohlen_pent_top.scl	BP Pentatonic, TOP tuning (245/243 tempered)
170	bohlen_pyth.scl	Cycle of 13 7/3 BP tenths
171	bohlen_quintuple_j.scl	Bohlen-Pierce quintuple scale (just version of 65ED3). Georg Hajdu (2017)
172	bohlen_quintuple_t.scl	Bohlen-Pierce quintuple scale, 65th root of 3. Georg Hajdu (2017)
173	bohlen_sirius.scl	Paul Erlich, Sirius-7, TOP tuning (3125/3087 tempered)
174	bohlen_t.scl	Bohlen, scale based on the twelfth
175	bohlen_t_ji.scl	Bohlen, scale based on twelfth, just version
176	bolivia.scl	Observed scale from pan-pipe from La Paz. 1/1=171 Hz
177	boomsliter.scl	Boomsliter & Creel basic set of their referential tuning. [1 3 5 7 9] x u[1 3 5] cross set
178	boop19.scl	19
179	cluster8e.scl	Eight-Tone Triadic Cluster 5:6:8
180	cluster8f.scl	Eight-Tone Triadic Cluster 5:8:6
181	cluster8h.scl	Eight-Tone Triadic Cluster 4:7:5, genus [5557]
182	cluster8i.scl	Eight-Tone Triadic Cluster 5:6:7
183	cluster8j.scl	Eight-Tone Triadic Cluster 5:7:6
184	cohenf_11.scl	Flynn Cohen, 7-limit scale of "Rameau's nephew" (1996)

185	coherent49.scl	Generator is the positive root of x^4 - x^2 - 1, Raph, Meta-Sidi, 72&121 temperament sqrtphi <30 35 38 39
186	coleman10.scl	Coleman 10 (2001)
187	coleman11.scl	Jim Coleman's XI piano temperament. TL 16 Mar 1999
188	coleman16.scl	Balanced 16 from Jim Coleman Sr. (2001)
189	coleman4.scl	Coleman IV from Jim Coleman Sr.
190	coll7.scl	7
191	collangettes.scl	d'Erlanger vol.5, p. 23. Père Maurice Collangettes, 24 tone Arabic system
192	collapsar.scl	An 11-limit patent val superwakalix
193	colonna1.scl	Colonna's irregular Just Intonation no. 1 (1618)
194	colonna2.scl	Colonna's irregular Just Intonation no. 2 (1618)
195	compton48.scl	Compton[48] 11-limit tweaked
196	concertina.scl	English Concertina, Helmholtz/Ellis, p. 470
197	cons11.scl	Set of intervals with num + den <= 11 not exceeding 2/1
198	cons12.scl	Set of intervals with num + den <= 12 not exceeding 2/1
199	cons13.scl	Set of intervals with num + den <= 13 not exceeding 2/1
200	cons14.scl	Set of intervals with num + den <= 14 not exceeding 2/1
201	cons15.scl	Set of intervals with num + den <= 15 not exceeding 2/1
202	cons16.scl	Set of intervals with num + den <= 16 not exceeding 2/1
203	cons17.scl	Set of intervals with num + den <= 17 not exceeding 2/1
204	cons18.scl	Set of intervals with num + den <= 18 not exceeding 2/1
205	cons19.scl	Set of intervals with num + den <= 19 not exceeding 2/1
206	cons20.scl	Set of intervals with num + den <= 20 not exceeding 2/1
207	cons21.scl	Set of intervals with num + den <= 21 not exceeding 2/1
208	cons8.scl	Set of intervals with num + den <= 8 not exceeding 2/1
209	cons9.scl	Set of intervals with num + den <= 9 not exceeding 2/1
210	cons_5.scl	Set of consonant 5-limit intervals within the octave
211	cons_7.scl	Set of consonant 7-limit intervals of tetrad 4:5:6:7 and inverse
212	cons_7a.scl	Set of consonant 7-limit intervals, harmonic entropy minima
213	cont_frac1.scl	Continued fraction scale 1, see McLaren in Xenharmonikon 15, pp.33-38
214	cont_frac2.scl	Continued fraction scale 2, see McLaren in Xenharmonikon 15, pp.33-38
215	corner11.scl	Quadratic Corner 11-limit. Chalmers '96
216	corner13.scl	Quadratic Corner 13-limit. Chalmers '96
217	corner17.scl	Quadratic Corner 17-limit.
218	corner17a.scl	Quadratic Corner 17 odd limit.
219	corner7.scl	Quadratic corner 7-limit. Chalmers '96
220	corner9.scl	First 9 harmonics of 5th through 9th harmonics
221	corners11.scl	Quadratic Corners 11-limit, John Chalmers (1996)
222	corners13.scl	Quadratic Corners 13-limit. Chalmers '96
223	corners7.scl	Quadratic Corners 7-limit. Chalmers '96
224	corrette.scl	Corrette temperament, modified 1/4-comma meantone
225	corrette2.scl	Michel Corrette, modified meantone temperament (1753)
226	corrette3.scl	Corrette's monochord (1753), also Marpurg 4 and Yamaha Pure Minor
227	coul_12.scl	Scale 1 5/4 3/2 2 successively split largest intervals by smallest interval
228	coul_12a.scl	Scale 1 6/5 3/2 2 successively split largest intervals by smallest interval
229	coul_12sup.scl	Superparticular approximation to Pythagorean scale. Op de Coul, 2003

230	coul_13.scl	Symmetrical 13-tone 5-limit JI scale
231	coul_17sup.scl	Superparticular approximation to Pythagorean 17-tone scale. Op de Coul, 2003
232	coul_20.scl	Tuning for a 3-row symmetrical keyboard, Op de Coul (1989)
233	coul_27.scl	Symmetrical 27-tone 5-limit just system, 67108864/66430125 and 25/24
234	counterschismic.scl	Counterschismic temperament, g=498.082318, 5-limit
235	couperin.scl	Couperin modified meantone
236	couperin_org.scl	F. Couperin organ temperament (1690), from C. di Veroli, 1985
237	cpak19a.scl	First 19-epimorphic ordered tetrad pack scale, Gene Ward Smith, TL 23-10-2005
238	cpak19b.scl	Second 19-epimorphic ordered tetrad pack scale, Gene Ward Smith, TL 23-10-2005
239	cross13.scl	13-limit harmonic/subharmonic cross
240	cross2.scl	John Pusey's double 5-7 cross reduced by 3/1
241	cross2_5.scl	double 3-5 cross reduced by 2/1
242	cross2_7.scl	longer 3-5-7 cross reduced by 2/1
243	cross3.scl	John Pusey's triple 5-7 cross reduced by 3/1
244	crossbone1.scl	7-limit Crossbone Scale (1st order, 1st sepent)
245	cross_7.scl	3-5-7 cross reduced by 2/1, quasi diatonic, similar to Zalzal's, Flynn Cohen
246	cross_72.scl	double 3-5-7 cross reduced by 2/1
247	cross_7a.scl	2-5-7 cross reduced by 3/1
248	cruciform.scl	Cruciform Lattice
249	cube3.scl	7-limit Cube[3] scale, Gene Ward Smith
250	cube4.scl	7-limit Cube[4] scale, Gene Ward Smith
251	cw12_11.scl	CalkinWilf(<12 19 28 34 42|)
252	cw19_11.scl	CalkinWilf(<19 30 44 53 66|)
253	cw19_5.scl	CalkinWilf(<19 30 44|)
254	cw19_7.scl	CalkinWilf(<19 30 44 53|)
255	cx4.scl	10

Uploading and downloading programs and banks

XFM2 distribution includes *XLoad*, a command-line utility (Windows) which allows to read and write program and bank files into from the device. A bank is composed of the 128 programs in the EEPROM memory.

The available commands, syntax and examples are displayed by typing *XLoad* and hitting *Enter*.

```
C:\windows\system32\cmd.exe

D:\>XLoad

Usage:  xload PORT OPTION [ # ] FILENAME

Options:
        -r              Read Bank file from device
        -w              Write Bank file to device
        -rp #           Read Program file from device
        -wp #           Write Program file to device
        -$              Initialize EEPROM (Program memory)
        -t              Load Tuning Definition File
        -img            Load Synthesizer Image


Examples:
        XLoad COM7 -r test.mem          (Reads all programs into 'test.mem')
        XLoad COM7 -rp 3 prog3.prg      (Reads program #3 into 'prog3.prg')
        XLoad COM4 -w dx_pianos.mem     (Writes bank file 'dx_pianos.mem' into device)
        XLoad COM4 -wp 3 my_dx.prg      (Writes program file 'my_dx.prg' into device's slot 3)


D:\>
```

MIDI implementation chart

XFM2 can receive and process the following MIDI messages:

Parameter	Range
NOTE-ON, NOTE-OFF NUMBERS	0-127
NOTE-ON VELOCITY	0-127
PROGRAM CHANGE	0-127
PITCH BEND	0-16383
CHANNEL AFTERTOUCH	0-127
ACTIVE SENSING	Active
MIDI CLOCK	Used to set Arpeggiator tempo, if internal tempo setting is 0.
CONTINUOUS CONTROLS	0-127
MODULATION WHEEL	CC 01
BREATH CONTROL	CC 02
FOOT CONTROL	CC 04
PORTAMENTO TIME	CC 05
VOLUME	CC 07
PAN	CC 10
FX PARAM 1: REVERB DECAY	CC 12
FX PARAM 2: DELAY TIME	CC 13
ARPEGGIATOR VELOCIY	CC 16
SUSTAIN PEDAL	CC 64
PERFORMANCE CONTROL 1	CC 70
PERFORMANCE CONTROL 2	CC 71
PERFORMANCE CONTROL 3	CC 72
PERFORMANCE CONTROL 4	CC 73
REVERB LEVEL	CC 91
TREMOLO LEVEL	CC 92 (Amplitude Modulation depth)
CHORUS LEVEL	CC 93
DELAY LEVEL	CC 94
PHASER LEVEL	CC 95
ALL SOUNDS OFF	CC 123
SYSTEM EXCLUSIVE MESSAGES	See "*Midi System Exclusive implementation chart"* for details.

MIDI System Exclusive implementation chart

All parameters in XFM2 can be controlled via MIDI System Exclusive messages.
The message format for the messages is as follows:

F0 43 UU P1 P2 V1 V2 F7

Where:

UU = Unit number (#0 or #1 for each XFM2 unit, or #7F for "all units").
P1 P2 = Parameter number, 0~511 in two hexadecimal 7-bit values (see *Calculating 7-bit values*).
V1 V2 = Value, 0~255 in two hexadecimal 7-bit values (see *Calculating 7-bit values*).

Calculating 7-bit values

To calculate the two 7-bit values for any number (V1 = MSB, V2 = LSB):

V1 and V2 must satisfy: *number = V1 * 128 + V2*

So, if the number is smaller than 128, then:

V1 = 00
V2 = number

And if the number is equal or greater than 128, then:

V1 = truncate(number / 128)
V2 = number – (V1 * 128)

Example: 499; V1 = 03H, V2 = 73H

Parameter numbers

OPERATOR	OP1	OP2	OP3	OP4	OP5	OP6
ALGO	1	2	3	4	5	6
FEEDBACK	7	8	9	10	11	12
SYNC	13					
PHASE	286	287	288	289	290	291
MODE	14					
RATIO	15	16	17	18	19	20
RATIO FINE	21	22	23	24	25	26
FINE	27	28	29	30	31	32
LEVEL	33	34	35	36	37	38
LEVEL L	256	258	260	262	264	266
LEVEL R	257	259	261	263	265	267
VELO SENS	39	40	41	42	43	44
KEY BP	45	46	47	48	49	50
KEY LDEPTH	51	52	53	54	55	56
KEY RDEPTH	57	58	59	60	61	62
KEY LCURVE	63	64	65	66	67	68
KEY RCURVE	69	70	71	72	73	74
L0	181	182	183	184	185	186
L1	75	76	77	78	79	80
L2	82	83	84	85	86	87
L3	89	90	91	92	93	94
L4	96	97	98	99	100	101
L5	193	194	195	196	197	198
DLY	187	188	189	190	191	192
R1	103	104	105	106	107	108
R2	110	111	112	113	114	115
R3	117	118	119	120	121	122
R4	124	125	126	127	128	129
R5	199	200	201	202	203	204
RATE KEY	140	141	142	143	144	145
EG LOOP	244					
EG LOOP SEG	245					
AMS	159	160	161	162	163	164
PMS	222	223	224	225	226	227
WAVE 1	236	237	238	239	240	241
WAVE 2	268	269	270	271	272	273
W MODE	274	275	276	277	278	279
W RATIO	280	281	282	283	284	285

PROGRAM

PITCH EG	
L0	205
L1	130
L2	131
L3	132
L4	133
L5	207
DLY	206
R1	134
R2	135
R3	136
R4	137
R5	208
RATE KEY	146
RANGE	138
VELO	139

LFO	
WAVE	153
SPEED	151
SYNC	152
FADE	154
DEPTH PITCH	149
DEPTH AMP	150

OTHER	
BEND UP	172
BEND DN	173
TRANSPOSE	174
VOLUME	180
PAN	221
VELO OFFSET	242
EG RESTART	246
LEGATO	228
PORTA MODE	229
PORTA TIME	230
TUNING	251
OUTPUT	411

MODULATIONS

PITCH LFO	
AFTER	157
WHEEL	155
BREATH	209
FOOT	211

AMP LFO	
AFTER	158
WHEEL	156
BREATH	210
FOOT	212

EG BIAS	
AFTER	213
WHEEL	214
BREATH	215
FOOT	216

PITCH	
AFTER	217
BREATH	218
FOOT	219
RND	220

ARPEGGIATOR	
MODE	450
TEMPO	451
MUL	453
OCTAVES	454

PERF CONTROLS	
CTL 1H	420
CTL 1L	421
CTL 2H	422
CTL 2L	423
CTL 3H	424
CTL 3L	425
CTL 4H	426
CTL 4L	427

EFFECTS

BITCRUSHER	
DEPTH	380

DECIMATOR	
DEPTH	370

FILTER	
LO	320
HI	321

CHORUS/FLANGER	
DRY	360
WET	361
MODE	362
SPEED	363
DEPTH	364
FEEDBACK	365
LR PHASE	366

PHASER	
DRY	310
WET	311
MODE	312
SPEED	314
DEPTH	313
FEEDBACK	315
OFFSET	316
STAGES	317
LR PHASE	318

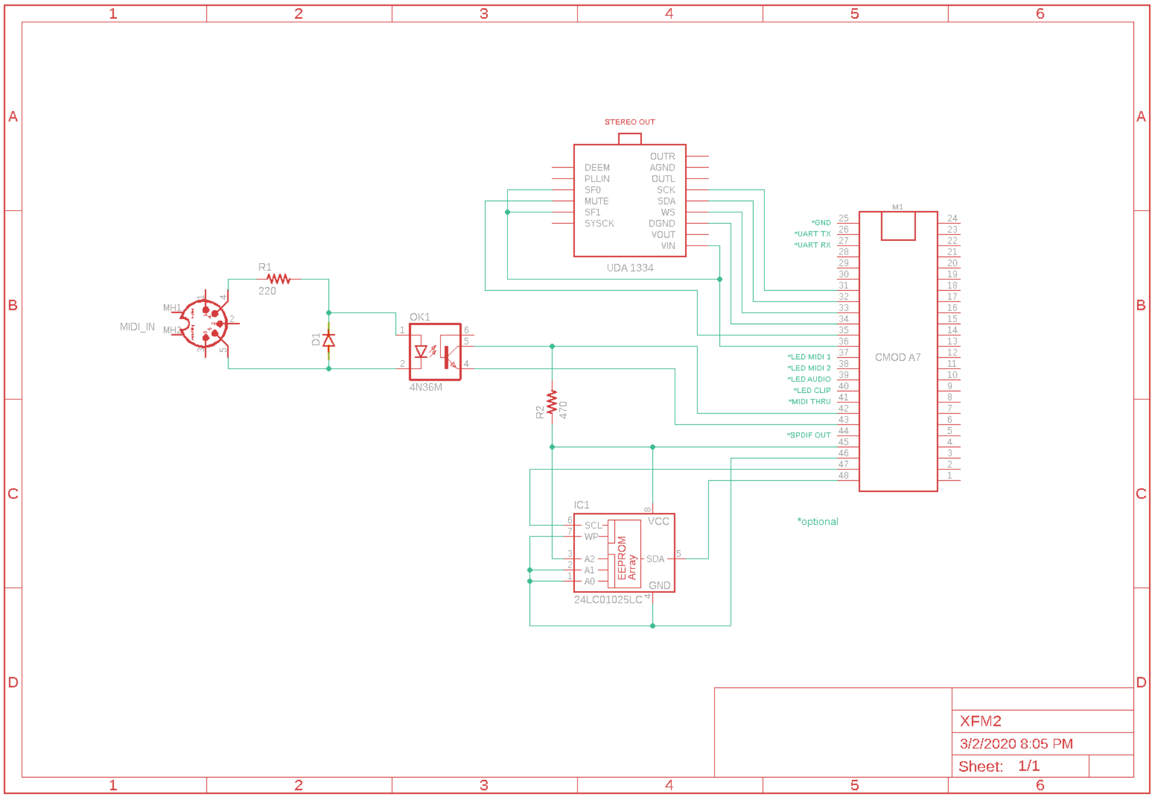
AM	
DEPTH	332
SPEED	330
RANGE	331
LR PHASE	333

DELAY	
DRY	300
WET	301
MODE	302
TIME	303
FEEDBACK	304
LO	305
HI	306
TEMPO	307
MUL	308
DIV	309

FX ROUTING	
ROUTING	410

REVERB	
DRY	390
WET	391
MODE	392
DECAY	393
DAMP	394

XFM2 Schematics



Bill of Materials

#	Qty	Component
1	1	Digilent CMOD A7 35T board ($89) ** make sure to pick the 35T version
2	1	Adafruit UDA 1334A board ($6.95)
3	1	24LC1025 I2C memory ($2.35)
4	1	4N36 photocoupler ($0.6)
5	1	50x70 prototyping board
6	1	1N4148 diode
7	1	220 ohm, 5%
8	1	470 ohm, 5%
9	1	5-pin female DIN connector

A brief note from the developer

Creating an FPGA-based synthesizer was a huge challenge.

As of the time of writing this, the available toolchains to synthesize and map HDL (Hardware Definition Language) are still years away from offering the productivity level of their equivalent for other technological solutions. In comparison, they are awkward, slow, poorly deterministic, and unreliable. It takes much longer to design, code, debug and document any feature.

FPGA devices, unlike DSP and MCU/SoC solutions, have almost nothing “turnkey”. Simple interfaces such as SPI, I2C, I2S, UART, SPDIF, SRAM, USB, Flash, DAC or PWM, and elementary functions such as clock dividers, counters, function and random number generators, must all be designed and implemented starting from scratch.

When rapid development is key, engineers have the option of resorting to high-level aggregation mechanisms. However, that wasn’t an option for XFM2, where each look-up table and flip-flop counted. Sometimes using behavioral description, sometimes describing features at a gate-level, doing FPGA code could be considered as an artform on itself.

On the other hand, once you see the results, it’s breathtaking. There’s almost nothing as satisfying in electronics builds as looking at a complete project in which you have designed *everything*, down to the last minimally accessory gate. It’s the mecca for control-freaks.

But this is only the beginning. An expanded satisfaction comes from seeing that your design is future proof, unlimitedly scalable and more portable than any other embedded system solution.

This advantage allowed me to complete XFM2 and release it few months after XFM. For a different FPGA, in a different board from another manufacturer, with a different toolchain. The block reusability proved satisfactory in this case.

It’s fascinating that eventually, the same design could be used to create an equivalent ASIC (the “real hardware” integrated circuits). I don’t find this particularly of interest, as having the chance to update a design in the field is much more attractive.

Once you have built XFM2, the same platform could be field-upgraded to run a completely different device, for instance a wavetable-based synthesizer, by simply uploading a different firmware.

XFM2 is the second generation of XFM. It expands XFM in every section, to the maximum area that fitted in the available silicon. Dozens of features had to be left behind in order to squeeze the design into the A7.

The good news is that most of those features are already done, and will find their way into a future project, built in a bigger FPGA (or, as FPGA prices won’t scale linearly, in multiple smaller FPGAs).

In the meantime, I hope you enjoy XFM2 as much as I do.
It’s a powerful machine, with lots of potential to help you find the best music inside of you.

Thank you again for building and using it. See you in the next gizmo!

/rgc