Tonal Plexus Software Instrument Support Guidelines

Aaron Andrew Hunt · Revision 7 · Jan 20, 2016

1. Basic Requirements

MIDI Notes and Channels

Tonal Plexus (TPX and U-PLEX) keyboards are built for microtonal music, and differ from standard keyboards by ...

- having 211 keys per octave instead of 12
- sending MIDI OUT on up to 16 MIDI channels

As a result of such a large number of keys per octave, the MIDI limitation of 128 notes per channel is already exceeded in a single keyboard octave, which leads to the second item, multichannel output. TPX multichannel output may be either two channels per octave (UNTUNED) or one channel per note (TUNED), as explained in section 2. Specifics concerning keyboard MIDI Note and Channel mapping can be found in the User Manuals available at **http://hpi.zentral.zone** but it is not necessary to know these details in order to provide basic software instrument compatibility.

The information given here applies to both TPX and U-PLEX keyboards. Details on the differences between these keyboards can be found at **http://hpi.zentral.zone**

Multitimbral Support

The Tonal Plexus is already compatible with multitimbral software synthesizers responding to basic MIDI messages without any special microtuning extensions. The multitimbral synthesizer must only support pitch bend. More specifically, the multitimbral software instrument ...

- **receives MIDI on *multiple arbitrary input channels* (any combination of 16 channels)**
- **supports Pitch Bend and Pitch Bend Range setting to 1 semitone on all channels**

For best compatibility, Pitch Bend support should be true 14-bit. The MIDI messages which set the pitch bend range are as follows:

Bank Change: (bank number 0-127)
Program Change: (patch number 0-127)
Controller 101: 0
Controller 100: 0
Controller 6: 1
Controller 38: 0

TPX sends each active channel the same bank and patch change messages, followed by pitch bend range setting controllers. Subsequent standard MIDI pitch bend and note messages instruct the instrument to produce microtonal music. The multichannel pitch bend method effectively reduces the multitimbral non-microtonal instrument to a monotimbral microtonal instrument.

The Monotimbral Solution

Instead of reducing the functionality of a multitimbral instrument, it would make more sense to expand the functionality of a monotimbral instrument. Because TPX sends MIDI out on multiple channels, multiple instances of a monotimbral software instrument are required in order to obtain polyphony from the instrument. This need not be the case, if the monotimbral instrument implements a special *multichannel receive mode*. The two ways TPX keyboards send MIDI out (TUNED and UNTUNED), result in two ways to make a software instrument compatible with TPX. Both methods involve receiving MIDI on multiple channels, *handling tuning and note messages independently per channel.*

2. Two Possible Methods

TPX UNTUNED OUTPUT - Multiple Tuning Tables Method

TPX UNTUNED output is sent to a single instance of a software instrument which …

- **receives MIDI on *all input channels* (16 channels)**
- **allows unique tuning tables for each channel, with the full MIDI frequency range output available for every MIDI note (any note = any pitch)**

According to this method, TPX sends standard MIDI Note messages, *and does not send any tuning messages*. Tuning tables are therefore required to be set up in advance within the software instrument. How this is done does not really matter, as long as the format is public so that an export method can be added to H-Pi software, which currently supports .SCL + .KBM, .TUN and .MSF. The best current standard to implement for multiple tuning tables is .MSF from AnaMark for which free C++ source code for translating files is available at **http://www.mark-henning.de/download_e.php#Tuning** Each instrument channel should respond to MIDI messages sent exclusively to that channel, within one instance of the software instrument.

Advantages:

- **polyphony is limited only by the software instrument engine**

This is an advantage for the user, but may present a problem for the software instrument developer. The additional channelization functions of the microtuning mode require that normal polyphony limitations should be spread out over all MIDI channels, so that only one engine of the software synthesizer is needed, and in microtuning mode the instrument does not consume more resources than it would normal mode. A simple solution would be a polyphony per channel equal to the number of voices in normal mode polyphony divided by 16. A more subtle solution would involve a dynamic voice allocation algorithm.

Disadvantages:

- **it is not GM compatible in terms of transportability**
- **tuning tables are necessary within the software instrument**
- **tuning data is not persistent in the MIDI stream**
- **sample-based instruments have some complications to deal with** *(see Section 3)*

Tuning tables in software instruments can lead to undesirable proliferation of incompatible native formats; this is already the case for the limited support of tuning tables now existing in the industry.[1] If tuning tables are used, the tuning data is set prior to note messages, so it is not persistent in the MIDI stream. The MIDI note data will not produce identical results when sent to various instruments, because each

[1] for example, Kontakt, Absynth, Reason, and FluidSynth all use their own incompatible tuning table formats.

instrument is required to be set up in advance with the tuning table. Because the tuning data is external to the MIDI stream, the MIDI stream becomes meaningless per se. An MTS message, the *Single Note Non-Real Time Tuning Change*, resolves this problem **http://www.midi.org/techspecs/midituning.php** but requires thirteen bytes to tune a single key, so that sending such a message before every Note message is relatively inefficient. Presently, H-Pi products do not send MTS messages.

TPX TUNED OUTPUT - Multi-Channel Pitch Bend Method

TPX tuned output is sent to a single instance of a software instrument which ...

- **receives MIDI on *multiple arbitrary input channels* (any combination of 16 channels)**
- **supports 14-bit pitch bend and pitch bend range setting to 1 semitone on all channels**

No tuning tables are required in the software instrument. *Each channel should respond to MIDI messages sent exclusively to that channel, within one instance of the software instrument.* Note that this is **not** the same as OMNI receive, where channel information is discarded. Channel information is essential, as pitch bend affects all the notes on a MIDI channel, and this implementation assumes the number of notes received on each channel is **one**. The retuning is done external to the software instrument, by sending pitch bend and note messages on user selected channels using a dynamic allocation algorithm (Patent Application US 11/824,243).

Advantages:

· **it is GM compatible in terms of transportability**
· **tuning tables are not necessary in the software instrument**
· **tuning data is persistent in the MIDI stream**
· **problems for sample-based instruments are bypassed**
·

Because the tuning tables are external to the software instrument, there is no need for a native tuning table format. Tuning data is persistent in the MIDI stream, so that it is directly meaningful, allowing direct editing, and so that it produces identical results when sent to various instruments.

Disadvantages:

· **polyphony is limited to 16 voices**

While this is a disadvantage for the user, for software instrument developers it is an advantage, because the additional channelization functions of the microtuning mode are far outbalanced by processing of fewer voices than normal mode polyphony. Thus in this microtuning mode, the software instrument should actually consume fewer processor intensive resources than in normal mode.
Refer to this method in your documentation as **General MIDI Microtuning**.

3. Sample-Based Instrument Concerns

The information in this section concerns sample-based instruments. The information may also be relevant for instruments based on synthesis.

Correct Sample-To-MIDI Note Mapping

Sample sets are assumed to be in twelve tone equal temperament (12ET). To maintain realistic instrument timbres, it is essential in microtuning samples that each sample be minimally retuned for each MIDI note. **A sample should never be retuned more than a quartertone above or below its original 12ET pitch.** The Pitch Bend method takes care of this automatically, but the multi-table method does not. Imagine the user wants to tune MIDI key 69, which is normally at A440Hz, to a flat C at 259Hz. The tuning table must remap the original sample for MIDI key 69 to C261Hz; **it must not retune the sample for A440Hz to 259Hz because this results in an unrecognizable timbre**. This point cannot be stressed enough. Software instruments which do not properly remap samples, such as the open source FluidSynth, are virtually useless for microtuning anything other than 12 tone temperaments.

Sample ON / OFF Access

A problem arises in arbitrarily microtuning 12ET sample sets when multiple MIDI notes are mapped to different tunings of the same original 12ET sample. This problem can arise in both pitch bend and multi-table methods. *If multi-channel pitch bend is used, samples must be available for multiple independent instances on different MIDI channels*. *If the multi-table method is used, samples must be available for multiple independent instances on the same MIDI channel*. For example, imagine that three arbitrary MIDI notes (the note numbers do not matter) are tuned to different versions of A4, for example A 438Hz, A 440Hz, and A 444Hz. The sample originally recorded at A440Hz, normally mapped to MIDI key 69, must be available to all three keys, and must be turned ON and OFF *independently* according to the MIDI note ON and OFF of each mapped MIDI key. For example, a NOTE ON message which calls the sample at A 438Hz *must not be stolen* by a NOTE ON message calling the sample at A 440 Hz. Likewise, it must not be turned off by a NOTE OFF message of any key calling the sample at any close frequency. Some sampler engines which provide retuning capabilities, such as Native Instruments Kontakt, do not allow samples to be accessed in this way, resulting in sounding pitches cutting out, and incapacity to play very closely tuned pitches using the same sample. Workarounds can be used which access samples further from the target pitch, but the resulting timbre for these widely retuned pitches is not ideal.

4. Implementation

Choosing a Method

Considering the advantages and disadvantages of the two methods outlined above, which method should be implemented?

For the end user, it is a case of ease-of-use versus polyphony. Pitch bend is easier for the user (of H-Pi products), but multiple tables gives the option of better polyphony and allows multitimbral instruments to remain multitimbral.

For the developer, it is a case of development cost versus operability and customer satisfaction. The pitch bend method provides simple implementation and basic monotimbral polyphonic compatibility for both multitimbral and monotimbral instruments, while the multiple table method is more specific and powerful but is more trouble to implement.

Weighing user needs against the development costs is likely to fall in favor of the pitch bend method, particularly when an instrument lacks pitch bend initially, considering that adding full pitch bend support fulfills part of the General MIDI standard requirement and benefits the widest audience. End users are likely to accept a limitation of 16 voices in order to gain microtuning capability with other ease-of-use advantages from a software instrument they already use. However, it is recommended that instruments already supporting pitch bend retuning should add support for multiple tables, giving the end user the choice between ease-of-use and maximum polyphony. Should development resources be available, adding support for both pitch bend and multi-table retuning is ideal for instruments which currently lack support for either method.

It should be emphasized that tuning involves many variables and can be quite confusing; therefore, ideally tuning data is explicit, is connected with note messages, and exists in the MIDI data stream at all times. Two advantages of this transparency are that any device receiving data starting from at any point will produce correct output, and an event list viewed by a human is immediately meaningful.

Compatibility Tables

Use the following tables to determine the compatibility status of your software instrument to help you decide on the best course of development action. As discussed above, multichannel input is required for basic compatibility, which should be at the very least *monotimbral* and *polyphonic*.

PITCH BEND (GM Microtuning) METHOD				
Instrument Type	**Receives on Multiple Channels**	**Supports Pitch Bend on Each Channel**	**Allows Pitch Bend Range to be Set on Each Channel**	**Compatibility Status**
Multitimbral	yes	no	no	Not Compatible
Multitimbral	yes	yes	no	Poor Monotimbral Polyphonic Compatibility
Multitimbral & Monotimbra l	**yes**	**yes**	**yes**	**Basic Monotimbral Polyphonic Compatibility**
Monotimbral	no	yes	yes	Monophonic Compatibility
Monotimbral	no	no	yes	Poor Monophonic Compatibility
Monotimbral	no	no	no	Not Compatible

MULTI-TABLE METHOD				
Instrument Type	**Receives on Multiple Channels**	**Allows Full MIDI Frequency Range to Every Note**	**Allows Each Channel to Have Its Own Tuning Table**	**Compatibility Status**
Multitimbral	yes	no	no	Not Compatible
Multitimbral	yes	yes	no	Poor Multitimbral Polyphonic Compatibility
Multitimbral	**yes**	**yes**	**yes**	**Basic Multitimbral Polyphonic Compatibility**
Monotimbra l	**yes**	**yes**	**yes**	**Basic Monotimbral Polyphonic Compatibility**
Monotimbral	no	yes	yes	LImited Monotimbral Polyphonic Compatibility
Monotimbral	no	no	yes	Poor Monotimbral Polyphonic Compatibility
Monotimbral	no	no	no	Not Compatible

Communication and Help

Please let me know how I can improve this document, and how I can best collaborate with you. I am happy to work with you to beta test your software. Please also visit the H-Pi Instruments website to download product user manuals, free software and other documentation.

Thanks,
Aaron Hunt

Aaron Andrew Hunt
H-Pi Instruments
http://hpi.zentral.zone
hpiinstruments@zentral.zone