MIDI Tuning Messages

Bank/Dump Extensions
Scale/Octave Extensions
GM-2 MIDI Tuning *(pdf)*

The MIDI Tuning specification allows the sharing of "microtunings" (user-defined scales other than 12-tone equal temperament) among instruments, and the switching of these tunings during real-time performance.

The MIDI Tuning messages include:

- Bulk Tuning Dump Request (non-real-time)
- Bulk Tuning Dump (non-real-time)
- Single-note Tuning Change (real-time)

Even though the first two messages are in the Universal Non-Real Time area and the last in the Real Time area, they keep the same sub-IDs to more obviously group them and possibly ease the parsing of them. Single Note Retuning allows retuning of individual MIDI note numbers to new frequencies in real time as a performance control.

The specification does not attempt to dictate how any manufacturer implements microtuning, but provides a general means of sharing tuning data among different instruments. This goal requires some shared assumptions which have some architectural implications. The standard requires that any of the 128 defined MIDI key numbers (or at least those MIDI key numbers covered by the instrument's playable range) be tunable to any frequency within the proposed frequency range. The standard also strongly suggests, but does not enforce, an exponential (constant cents) rather than linear (constant Hertz) tuning resolution across the instrument's frequency range.

The specification permits the changing of tunings in real-time, both by the selection of presets and on a per-note basis. When a sounding note is affected by either real-time tuning message, the note should instantly be re-tuned to the new frequency while it continues to sound; this change should occur without glitching, forced Note-Offs, re-triggering or other audible artifacts (see section 4, "Additional").

The specification provides for 128 tuning memory locations (programs). As with the MIDI program change message, this is a maximum value. An instrument supporting MIDI Tuning may have any lesser number of tuning programs. The specification requires only that all existing tuning programs respond to the messages as specified (See section 3, "Continuous Controller Messages").

Although directly applicable to some existing instruments, the specification attempts to define a coherent framework within which the designers of future instruments can profitably work. It is hoped that by providing this framework the specification will make microtunability more easily implemented and more common on MIDI instruments.

Frequency Data Format

The frequency resolution of MIDI Tuning should be stringent enough to satisfy most demands of music and experimentation. The specification provides resolution somewhat finer than one-hundredth of a cent. Instruments may support MIDI tuning without necessarily providing this resolution in their hardware; the specification simply permits the transfer of tuning data at any resolution up to this limit.

Frequency data shall be sent via system exclusive messages. Because system exclusive data bytes have their high bit set low, containing 7 bits of data, a 3-byte (21-bit) frequency data word is used for specifying a frequency with the suggested resolution. An instrument which does not support the full suggested resolution may discard any unneeded lower bits on reception, but it is preferred where possible that full resolution be stored internally, for possible transmission to other instruments which can use the increased resolution.

Frequency data shall be defined in units which are fractions of a semitone. The frequency range starts at MIDI note 0, C = 8.1758 Hz, and extends above MIDI note 127, G = 12543.875 Hz. The first byte of the frequency data word specifies the nearest equal-tempered semitone below the frequency. The next two bytes (14 bits) specify the fraction of 100 cents above the semitone at which the frequency lies. Effective resolution = 100 cents / 214 = .0061 cents.

One of these values (7F 7F 7F) is reserved to indicate not frequency data but a "no change" condition. When an instrument receives these bytes as frequency data, it should make no change to its stored frequency data for that MIDI key number. This is to prevent instruments which do not use the full range of 128 MIDI key numbers from sending erroneous tuning data to instrument which do use the full range. The three-byte frequency representation may be interpreted as follows:

0xxxxxxx 0abcdefg 0hijklmn

xxxxxxx = semitone
abcdefghijklmn = fraction of semitone, in .0061-cent units

Examples of frequency data:

00 00 00 = 8.1758 Hz (C – normal tuning of MIDI key no. 0)
00 00 01 = 8.2104 Hz
01 00 00 = 8.6620 Hz
0C 00 00 = 16.3516 Hz
3C 00 00 = 261.6256 Hz (middle C)
3D 00 00 = 277.1827 Hz (C# – normal tuning of MIDI key no. 61)
44 7F 7F = 439.9984 Hz

45 00 00 = 440.0000 Hz (A-440)
45 00 01 = 440.0016 Hz
78 00 00 = 8372.0190 Hz (C – normal tuning of MIDI key no. 120)
78 00 01 = 8372.0630 Hz
7F 00 00 = 12543.8800 Hz (G – normal tuning of MIDI key no. 127)
7F 00 01 = 12543.9200 Hz
7F 7F 7E = 13289.7300 Hz (top of range)
7F 7F 7F = no change (reserved)

[BULK TUNING DUMP REQUEST]

A bulk tuning dump request is as follows:

F0 7E <device ID> 08 00 tt F7

F0 7E	Universal Non-Real Time SysEx header
<device ID>	ID of target device
08	sub-ID#1 (MIDI Tuning)
00	sub-ID#2 (bulk dump request)
tt	tuning program number (0 – 127)
F7	EOX

The receiving instrument shall respond by sending the bulk tuning dump message described in the following section for the tuning number addressed.

[BULK TUNING DUMP]

A bulk tuning dump comprises frequency data in the 3-byte format outlined in section 1, for all 128 MIDI key numbers, in order from note 0 (earliest sent) to note 127 (latest sent), enclosed by a system exclusive header and tail. This message is sent by the receiving instrument in response to a tuning dump request.

F0 7E <device ID> 08 01 tt <tuning name> [xx yy zz] ... chksum F7

F0 7E	Universal Non-Real Time SysEx header
<device ID>	ID of responding device
08	sub-ID#1 (MIDI Tuning)
01	sub-ID#2 (bulk dump reply)
tt	tuning program number (0 – 127)
<tuning name>	16 ASCII characters
[xx yy zz]	frequency data for one note (repeated 128 times)
chksum	checksum (XOR of 7E <device ID> 01 tt <388 bytes>)***see "Checksum Calculation", below**
F7	EOX

If an instrument does not use the full range of 128 MIDI key numbers, it may ignore data associated with un-playable notes on reception, but it is preferred where possible that the full 128-key tuning be stored internally, for possible transmission to other instruments which can use the increased resolution. On transmission, it may if necessary pad frequency data associated with un-playable notes with the "no change" frequency data word defined above. For keys in the instrument's key range, the pitch that is sent should be the pitch that key would play if it were received as part of a note-on message. For keys outside the key range, 7F 7F 7F may be sent.

[SINGLE NOTE TUNING CHANGE (REAL-TIME)]

The single note tuning change message (Exclusive Real Time sub-ID#1 = 08) permits on-the-fly adjustments to any tuning stored in the instrument's memory. These changes should take effect immediately, and should occur without audible artifacts if any affected notes are sounding when the message is received.

F0 7F <device ID> 08 02 tt ll [kk xx yy zz] F7

F0 7F	Universal Real Time SysEx header
<device ID>	ID of target device
08	sub-ID#1 (MIDI Tuning)
02	sub-ID#2 (note change)
tt	tuning program number (0 – 127)
ll	number of changes (1 change = 1 set of [kk xx yy zz])
[kk]	MIDI key number
[xx yy zz]	frequency data for that key (repeated 'll' number of times)
F7	EOX

This message also permits (but does not require) multiple changes to be embedded in one message, for the purpose of maximizing bandwidth. The number of changes following is indicated by the byte ll; the total length of the message equals 8 + (ll x 4) bytes.

If an instrument does not support the full range of 128 MIDI key numbers, it should ignore data associated with un-playable notes on reception.

This message can be used to make changes in inactive (background) tunings as well. This message may also, at the discretion of the manufacturer, be transmitted by the instrument under particular circumstances (for example, while holding down one or more keys and pressing a "send-single-note-tuning" front panel button).

Changing Tuning Programs

Registered parameter numbers 03 and 04 are used to select any of the instrument's stored tunings as the "current" or active tuning. Instruments which permit the storage of multiple microtunings should respond to these messages by instantly changing the "current" tuning to the specified stored tuning. This change takes effect immediately and must occur without audible artifacts (notes-off, resets, re-triggers, glitches, etc.) if any affected notes are sounding when the message is received.

As with the MIDI program change message, no assumptions are made as to the underlying architecture of the instrument. For instance, in cases where layered or multi-timbral sounds might be assigned to different tunings, so that more than one tuning might be active, the manufacturer may decide how best to interpret these messages. The basic channel number might prove useful in discriminating between multiple active tunings, or a certain range of tuning programs might be set aside and defined as active.

The Tuning Program Select message is sent as a Registered Parameter number 03 controller message, followed by either a data entry, data increment, or data decrement controller message, e.g. (with running status shown):

Bn 64 03 65 00 06 tt (data entry)
Bn 64 03 65 00 60 7F (data increment)
Bn 64 03 65 00 61 7F (data decrement)

n = basic channel number
tt = Tuning Program number (1-128)

Likewise, the Tuning Bank Select Registered Parameter number 04 is sent as follows:

Bn 64 04 65 00 06 tt (data entry)
Bn 64 04 65 00 60 7F (data increment)
Bn 64 04 65 00 61 7F (data decrement)

n = basic channel number
tt = Tuning Bank number (1-128)

For maximum flexibility, this Bank Number is kept separate from the normal Program Change Bank Select (controller #00). However, an instrument may wish to link the two as a feature for the user, especially if a tuning bank is stored alongside a patch parameter bank (for example, on a RAM cartridge).

If an instrument receives a Tuning Program or Bank number for which it has no Program or Bank, it should ignore that message. Standard mappings of "common" tunings to program numbers are not being proposed at this time.

Comments:

There is some question as to whether instantaneous response to real-time tuning changes is desirable in every circumstance. In some performance situations it makes more sense if a tuning change affect only those notes which occur subsequent to the change, and not affect sounding notes. But there are also situations in which tuning changes should take place instantaneously, as specified in the standard, and should affect sounding notes without disrupting their continuity.

If the instrument responds well in the latter situation, some work-around is possible for the former. The reverse is not true. Therefore the standard requires that tuning changes immediately affect sounding notes. Manufacturers might, however, consider implementing a switchable "instantaneous/next-note-on" option within an instrument.

Single Note Retuning is intended for performance. Because of there are two primary concerns: 1) the RAM required for temporary copies of tuning tables; and 2) the computational load of smoothly updating the pitch of affected active notes. It is clear that in order to recognize the Single Note Retune message, a copy of the current Tuning Program needs to be kept in RAM. In a multi-timbral environment there is potentially a copy for each virtual instrument. A high-end instrument could afford the upwards of 8K of RAM needed for per-virtual-instrument copies. More modest instruments may choose to only implement one alterable RAM table and either make it available only to the basic channel virtual instrument or require that all instruments share the same tuning. Provided that it is explained in the user's manual, any of these methods is acceptable.

MIDI Tuning Bank/Dump Extensions (CA-020)

The first three messages in these extensions are based on existing MIDI Tuning messages, with the addition of the bank select byte. In the MIDI Spec V4.2, document version 96.1, page 50, a Tuning Bank Select Registered Parameter Number (RPN) is defined for choosing among banks of tuning programs. However, no way to store a tuning message in a specific

bank is defined, because the messages do not include a bank number in addition to the tuning program number. These extensions correct that oversight.

The fourth new message provides a **non real-time** version of the Single Note Tuning message. The intent of a non-real time message is that it should specifically be sent as a setup message. If it is NOT sent as a setup message (that is, if it is sent during performance), it is assumed that the message will be ignored for notes that are already sounding. However, certain Recommended Practices, (RPs), may implement this differently. Manufacturers should therefore consult the specific RP for the required response and should document the device's response in the user manual.

The last two new messages support Scale/Octave tables (see Scale/Octave Tuning).

[BULK TUNING DUMP REQUEST (BANK)]

F0 7E 08 03 bb tt F7

F0 7E	Universal Non-Real Time SysEx header
<device ID>	ID of target device (7F = all devices)
08	sub-ID#1 = "MIDI Tuning Standard"
03	sub-ID#2 = "BULK TUNING DUMP REQUEST (BANK)"
bb	bank: 0-127 (described as 1-128 in MIDI Tuning Specification)
tt	tuning preset number: 0-127
F7	EOX

This message is identical to the BULK TUNING DUMP REQUEST except for the addition of the bank select byte (bb). The receiving device will send back a Tuning Dump message which will vary depending on which tables, (key-based or scale/octave), are supported by the instrument.

[KEY-BASED TUNING DUMP]

F0 7E <device ID> 08 04 bb tt <tuning name> [xx yy zz] ... chksum F7

F0 7E	Universal Non-Real Time SysEx header
<device ID>	ID of target device (7F = all devices)
08	sub-ID#1 = "MIDI Tuning Standard"
04	sub-ID#2 = "KEY-BASED TUNING DUMP"
bb	bank: 0-127 (described as 1-128 in MIDI Tuning Specification)
tt	tuning preset number: 0-127
<tuning name>	16 ASCII characters
[xx yy zz]	frequency data for one note (repeated 128 times)
checksum	see "Checksum Calculation", below
F7	EOX

This message is identical to the current BULK TUNING DUMP except for the addition of the bank select byte (bb). The message was renamed KEY-BASED TUNING DUMP to differentiate between it and the new scale/octave tuning messages. (See below)

[SINGLE NOTE TUNING CHANGE (REAL-TIME) (BANK)]

F0 7F <device ID> 08 07 bb tt ll [kk xx yy zz] ... F7

F0 7F	Universal Real-Time SysEx header
<device ID>	ID of target device (7F = all devices)
08	sub-ID#1 = "MIDI Tuning Standard"
07	sub-ID#2 = "SINGLE NOTE TUNING CHANGE (REAL-TIME)(BANK)"

bb
bank: 0-127
(described as 1-128 in MIDI Tuning Specification)

tt tuning preset number: 0-127

ll
number of changes
(1 change = 1 set of [kk xx yy zz])

[kk MIDI key number
xx yy zz] frequency data for that key
(repeated 'll' number of times)

F7 EOX

This is identical to the current SINGLE NOTE TUNING CHANGE (REAL-TIME) except for the addition of the bank select byte (bb). This message WILL affect currently sounding notes.

[SINGLE NOTE TUNING CHANGE (NON REAL-TIME) (BANK)]

F0 7E <device ID> 08 07 bb tt ll [kk xx yy zz] ... F7

F0 7E Universal Non-Real Time SysEx header

<device ID> ID of target device (7F = all devices)

08 sub-ID#1 = "MIDI Tuning Standard"

07 sub-ID#2 = "SINGLE NOTE TUNING CHANGE (NON REAL-TIME)(BANK)"

bb
bank: 0-127 (described as 1-128 in MIDI Tuning Spec)

tt tuning preset number: 0-127

ll
number of changes (1 change = 1 set of
[kk xx yy zz]) [kk MIDI key number
xx yy zz] frequency data for that key
(repeated 'll'number of times)

F7 EOX

This is identical to the current SINGLE NOTE TUNING CHANGE (REAL-TIME) except for the addition of the bank select byte (bb) and the change to a NON REAL -TIME header. This message allows the sender to specify a new tuning change that will NOT update the currently sounding notes.

[SCALE/OCTAVE TUNING DUMP, 1 byte format]

F0 7E <device ID> 08 05 bb tt <tuning name> [xx] ... chksum F7

F0 7E Universal Non-Real Time SysEx header

<device ID> ID of target device (7F = all devices)

08 sub-ID#1 = "MIDI Tuning Standard"

05 sub-ID#2 = "SCALE/OCTAVE TUNING DUMP, 1 byte format"

bb
bank: 0-127 (described as 1-128 in MIDI Tuning Spec)

tt tuning preset number: 0-127

<tuning name> 16 ASCII characters

[xx]
frequency data for C,C#,... B (12 bytes total)

00H means -64 Cent
40H means +/- 0 Cent
7FH means +63 Cent

chksum see "Checksum Calculation", below

F7	EOX

[SCALE/OCTAVE TUNING DUMP, 2 byte format]

F0 7E <device ID> 08 06 bb tt <tuning name> [xx yy] ... chksum F7

F0 7E	Universal Non-Real Time SysEx header
<device ID>	ID of target device (7F = all devices)
08	sub-ID#1 = "MIDI Tuning Standard"
06	sub-ID#2 = "SCALE/OCTAVE TUNING DUMP, 2 byte format"
bb	bank: 0-127 (described as 1-128 in MIDI Tuning Spec)
tt	tuning preset number: 0-127
<tuning name>	16 ASCII characters
[xx yy]	frequency data for C,C#,... B (24 bytes total) 00H 00H means -100 cents (8,192 steps of .012207 cents) 40H 00H means 0 cents (equal temperament) 7FH 7FH means +100 cents (8,191 steps of .012207 cents)
chksum	see "Checksum calculation", below
F7	EOX

The minimum and maximum offsets are approximately 100 cents.

Checksum Calculation:

The "Dump" messages all include a checksum field. Due to ambiguities in the instructions for the original Bulk Dump Message (Sub-ID#2="01") various manufacturers have implemented that checksum differently, and it is now recommended that receivers may ignore the checksum in that message. For all other Dump messages the checksum field is calculated by successively XOR'ing the bytes in the message, excluding the F0, F7, and the checksum field... The resulting value is then AND'ed with 7F, to create a 7 bit value.

MIDI Tuning Scale/Octave Extensions (CA-021/RP-020)

Scale/Octave Tuning is micro-tuning that is automatically repeated in every octave by calibrating a single octave of notes in small fractions of an equal-tempered half-step. The original MIDI tuning dump message had to define a frequency to each of 128 keys. This proposal defines an easier micro tuning that sets offsets from an equal-tempered half-step by the cent.

If the instrument has a selectable octave tuning presets, this message will offset the tuning from the currently selected preset. The default behavior is to assume that the instrument is set to an equal temperment, and the message will offset from that. Each message that is received will offset from the original preset, not from the modified tuning.

Scale/Octave Tuning consists of the following four messages (*plus the two new Scale/Octave Tuning Dump messages described in MIDI Tuning Bank/Dump Extensions, above*). The first and second messages will offset the tuning of each note in cents. The range is -64/+63 cents. The first message is a real time version. Notes that are already sounding should be updated after receiving this message, and future notes will be tuned to the new offsets. The second message is non-real time. The third and fourth messages will offset the tuning of each note using a 2 byte encoding, which allows greater range and resolution. With two tuning bytes, the tuning range can be expanded to approximately +/-100 cents with a resolution of .012207 cents (200 cents divided by 16,384 = .012207 cents). The third message is a real time message. Notes that are already sounding should be updated after receiving this message, and future notes will be tuned to the new offsets. The fourth message is non real-time.

A message can update multiple MIDI channels simultaneously. This is achieved by using a 3 byte channel field in the SysEx message, with 1 bit representing each MIDI channel. There are 5 additional bits in one of the channel bytes, which must be set to 0. These bits are reserved for future expansion of the tuning messages. They are NOT to be used in any proprietary fashion.

[SCALE/OCTAVE TUNING 1-BYTE FORM (REAL-TIME)]

F0 7F <device ID> 08 08 ff gg hh [ss] ... F7

F0 7F	Universal Real-Time SysEx header
<device ID>	ID of target device (7F = all devices)

08	sub-ID#1 = "MIDI Tuning Standard"
08	sub-ID#2 = "scale/octave tuning 1-byte form (Real-Time)"
ff	channel/options byte 1 bits 0 to 1 = channel 15 to 16 bit 2 to 6 = reserved for future expansion
gg	channel byte 2 - bits 0 to 6 = channel 8 to 14
hh	channel byte 3 - bits 0 to 6 = channel 1 to 7
[ss]	12 byte tuning offset of 12 semitones from C to B 00H means -64 cents 40H means 0 cents (equal temperament) 7FH means +63 cents
F7	EOX

[SCALE/OCTAVE TUNING 1-BYTE FORM (NON REAL-TIME)]

F0 7E <device ID> 08 08 ff gg hh [ss] ... F7

F0 7E	Universal Non Real-Time SysEx header
<device ID>	ID of target device (7F = all devices)
08	sub-ID#1 = "MIDI Tuning Standard"
08	sub-ID#2 = "scale/octave tuning 1-byte form (Non Real-Time)"
ff	channel/options byte 1 bits 0 to 1 = channel 15 to 16 bit 2 to 6 = reserved for future expansion
gg	channel byte 2 - bits 0 to 6 = channel 8 to 14
hh	channel byte 3 - bits 0 to 6 = channel 1 to 7
[ss]	12 byte tuning offset of 12 semitones from C to B 00H means -64 cents 40H means 0 cents (equal temperament) 7FH means +63 cents
F7	EOX

[SCALE/OCTAVE TUNING 2-BYTE FORM (REAL-TIME)]

F0 7F <device ID> 08 09 ff gg hh [ss tt] ... F7

F0 7F	Universal Real-Time SysEx header
<device ID>	ID of target device (7F = all devices)
08	sub-ID#1 = "MIDI Tuning Standard"
09	sub-ID#2 = "scale/octave tuning 2-byte form (Real-Time)"
ff	channel/options byte 1 bits 0 to 1 = channel 15 to 16 bit 2 to 6 = reserved for future expansion
gg	channel byte 2 - bits 0 to 6 = channel 8 to 14
hh	channel byte 3 - bits 0 to 6 = channel 1 to 7
[ss tt]	

	24 byte tuning offset of 2 bytes per semitone from C to B 00H 00H means -100 cents (8,192 steps of .012207 cents) 40H 00H means 0 cents (equal temperament) 7FH 7FH means +100 cents (8,191 steps of .012207 cents)
F7	EOX

[SCALE/OCTAVE TUNING 2-BYTE FORM (NON REAL-TIME)]

F0 7E <device ID> 08 09 ff gg hh [ss tt] ... F7

F0 7E	Universal Non Real-Time SysEx header
<device ID>	ID of target device (7F = all devices)
08	sub-ID#1 = "MIDI Tuning Standard"
09	sub-ID#2 = "scale/octave tuning 2-byte form (Non Real-Time)"
ff	channel/options byte 1 bits 0 to 1 = channel 15 to 16 bit 2 to 6 = reserved for future expansion
gg	channel byte 2 - bits 0 to 6 = channel 8 to 14
hh	channel byte 3 - bits 0 to 6 = channel 1 to 7
[ss tt]	24 byte tuning offset of 2 bytes per semitone from C to B 00H 00H means -100 cents (8,192 steps of .012207 cents) 40H 00H means 0 cents (eq5al temperament) 7FH 7FH means +100 cents (8,191 steps of .012207 cents)
F7	EOX

Notes:

Using a channel bitmap scheme in a system exclusive message is unconventional, but this ability is really necessary for the computer based tuning schemes, in order to change intonation identically on several channels quickly. (Ed: This text was amended 12/2000 by RP-025/amd1)

The intent of a non-real time message is that it should specifically be sent as a setup message. If it is NOT sent as a setup message (that is, if it is sent during performance), it is assumed that the message will be ignored for notes that are already sounding. However, certain Recommended Practices, (RPs), may implement this differently. Manufacturers should therefore consult the specific RP for the required response and should document the device's response in the user manual.

Recommended Practice (RP-020)
Defaults for Scale/Octave Tuning

If tuning presets are not supported by the instrument, it is assumed that the initial tuning of the instrument is equal temperament. If presets are supported, it is suggested that the first preset, selected by Bank 0H and Preset 0H, would be equal temperament. Tuning adjusters should begin by selecting Bank 0H, Preset 0H in order to start from equal temperament, if that is the desired behavior.
