Guide to the MIDI Software Specification

Contents

Overview

MIDI data is a serial stream of bytes (31,250 baud, asynchronous, 1 start bit, 1 stop bit) representing MIDI commands. A MIDI command comprises a **status byte** followed by a variable number of **data bytes** (possibly none).

Page content last updated : Mon 26 Sep 2022

Status bytes are identified due to their top bit being set. Consequently data bytes contain 7-bit values (0-127).

There are two main types of status bytes : **Channel** messages and **System** messages.

Channel messages use the lower 4 bits of the status byte to address 1 of 16 MIDI channels and the upper 4 bits to encode the specific message. A device can receive messages on more than one channel (known as *voice channels*), dependant on its *mode* of operation. However, there is a particular channel referred to as its *base channel* upon which it receives commands about which *mode* it should operate in and which voice (sound) it should use. Channel messages are sub-divided into two types :

- **Channel Voice** messages (&80 - &EF) allow control of devices dependant on what MIDI channel they are set to (i.e. their *base channel*). That is, they allow control of individual voices.
- **Channel Mode** messages (a subset of &Bn) are sent on a device's base channel, and are used to specify how a device should respond to Voice messages.

The **System** messages are not (generally) targeted for specific MIDI channels and are hence global messages. They are further divided into three types :

- **System Exclusive** messages (&F0) are variable length messages. There is possibly more complexity in the various messages provided by this single status byte than by the rest of the MIDI specification.
  It has become the main route for expansion of the standard. It provides for manufacturer defined, device-specific messages, and any other messages involving larger amounts of data (i.e. blocks of data rather than the odd byte or two) than allowed for in all other messages.
- **System Common** messages (&F1 - &F7) are involved with various system setup type operations, including sequencer (song) control and timing, and an instruction for a sound generator to ensure that it is *in tune*.
- **System Real Time** messages (&F8 - &FF) are all single byte commands involved primarily with the control and timing of sequencer (song) playback.

The main difference between System Common and System Real Time messages is that the latter can interleave other MIDI data. That is, Real Time messages can occur in the middle of *any* other message. This is due to their use in sequence synchronisation, and aids timing accuracy.

Summary of Status Bytes

Status byte		Number of data bytes	Description
Hex	**Binary**		
Channel Voice Messages			
8n	1000nnnn	2	Note Off
9n	1001nnnn	2	Note On (a velocity of zero = Note Off)
An	1010nnnn	2	Polyphonic Key Pressure (Aftertouch)
Bn	1011nnnn	2	Controller First data byte = 0-119
Cn	1100nnnn	1	Program Change (i.e. instrument/voice selection)
Dn	1101nnnn	1	Channel Pressure (Aftertouch)
En	1110nnnn	2	Pitch Bend
Channel Mode Messages			
Bn	1011nnnn	2	Select Channel Mode First data byte = 120-127

Page content last updated : Mon 26 Sep 2022

System Messages			
F0	11110000	<variable>	System Exclusive
F1-F7	11110xxx	0 to 2	System Common (xxx = 1-7)
F8-FF	11111xxx	0	System Real Time (xxx = 0-7)

Notes :

nnnn 0-15 = MIDI channel 1-16.

Running Status

Running Status is a data-thinning technique. It allows for the omission of status bytes if the current message to be transmitted has the same status byte (i.e. the same command and MIDI channel) as the previous message. It thus only applies to **Channel** (Voice and Mode) messages (0x8n - 0xEn).

System Real Time messages (0xF8 - 0xFF) are all single byte messages, and so running status is not applicable. Also, they can be interleaved within other messages without affecting running status in any way.

System Common and **System Exclusive** messages (0xF0 - 0xF7), on the other hand, cancel any running status, so *any* message following one of these messages *must* contain a status byte. As with System Real Time messages, running status is not applicable to System Common and System Exclusive messages, so these messages must *always* contain a status byte.

E.g. to play the C Major chord based on middle C :

Without running status :

0x90 0x3C 0x7F Note On for C4, max velocity
0x90 0x40 0x7F Note On for E4, max velocity
0x90 0x43 0x7F Note On for G4, max velocity

With running status, the second and third status bytes are omitted :

0x90 0x3C 0x7F Note On for C4, max velocity
0x40 0x7F Note On for E4, max velocity
0x43 0x7F Note On for G4, max velocity

Both approaches are valid, and so either may be used.

Coding issues

To be able to cope with running status, a receiving device should always remember the last status byte that it received (except for **System** messages), and if it doesn't receive a status byte when expected (on subsequent messages), it should assume that it's dealing with a running status situation. Apart from **System Real Time** messages, new status bytes will always force a receiver to adopt that new status, even if the previous message was incomplete.

A device that generates MIDI messages should always remember the last status byte that it sent (except for **System Real Time** messages), and if it needs to send another message with the same status (**Channel** messages only, not **System** messages), the status byte may be omitted.

Note messages

Running status is the reason behind the velocity = 0 variant of the Note On message (which acts as

a Note Off).

E.g. to play the same C Major chord as above (though also turning it off this time) :

Without running status :

0x90 0x3C 0x7F	Note On for C4, max velocity
0x90 0x40 0x7F	Note On for E4, max velocity
0x90 0x43 0x7F	Note On for G4, max velocity
. . . . .	Wait for duration of the chord ...
0x80 0x3C 0x3F	Note Off for C4, middle release velocity
0x80 0x40 0x3F	Note Off for E4, middle release velocity
0x80 0x43 0x3F	Note Off for G4, middle release velocity

With running status, and using Note On with velocity = 0, to act as Note Offs :

0x90 0x3C 0x7F	Note On for C4, max velocity
0x40 0x7F	Note On for E4, max velocity
0x43 0x7F	Note On for G4, max velocity
. . . . .	Wait for duration of the chord ...
0x3C 0x00	Note Off for C4, no release velocity information
0x40 0x00	Note Off for E4, no release velocity information
0x43 0x00	Note Off for G4, no release velocity information

Channel Voice Messages

The status byte of all Channel Voice messages is nibblised, with the top nibble (i.e. the top 4 bits) indicating the command (0x8 to 0xE), and the lower nibble indicating the MIDI channel.

The first three of these messages specify a MIDI note number : 0 - 127 (C-2 - G8), with middle C (C3) being note number 60. The standard 5 octave synthesizer keyboard range is 36 - 96, whereas the 88-note piano keyboard range is 21 - 108. Refer to this ready reckoner table of MIDI note numbers (../basic/notes.html) for the complete mapping.

8n : Note Off

3 bytes	`1000nnnn , 0kkkkkkk , 0vvvvvvv`
`1000nnnn`	Note Off status byte; nnnn (0-15) = MIDI channel 1-16
`kkkkkkk`	Note number (0-127)
`vvvvvvv`	Key Off (release) velocity (0-127); Defaults to 64 in the absence of velocity sensors

9n : Note On

3 bytes	`1001nnnn , 0kkkkkkk , 0vvvvvvv`
`1001nnnn`	Note On status byte; nnnn (0-15) = MIDI channel 1-16
`kkkkkkk`	Note number (0-127)
`vvvvvvv`	Key On (attack) velocity (1-127, soft-loud); 0 = Note Off; Defaults to 64 in the absence of velocity sensors

An : Polyphonic Key Pressure (Aftertouch)

Page content last updated : Mon 26 Sep 2022

3 bytes	1010nnnn , 0kkkkkkk , 0vvvvvvv
1010nnnn	Polyphonic Key Pressure status byte; nnnn (0-15) = MIDI channel 1-16
kkkkkkk	Note number (0-127)
vvvvvvv	Pressure value (0-127)

Bn : Controller

3 bytes	1011nnnn , 0ccccccc , 0vvvvvvv
1011nnnn	Controller change status byte; nnnn (0-15) = MIDI channel 1-16
ccccccc	Controller number (0-119) Values 120-127 are reserved for Channel Mode messages
vvvvvvv	Controller value (0-127); For switch controllers : 0 = Off, 127 = On

Cn : Program Change

2 bytes	1100nnnn , 0ppppppp
1100nnnn	Program Change status byte; nnnn (0-15) = MIDI channel 1-16
ppppppp	Program number (0-127)

Dn : Channel Pressure (Aftertouch)

2 bytes	1101nnnn , 0vvvvvvv
1101nnnn	Channel Pressure status byte; nnnn (0-15) = MIDI channel 1-16
vvvvvvv	Pressure value (0-127)

En : Pitch Bend

3 bytes	1110nnnn , 0vvvvvvv , 0hhhhhhh	
1110nnnn	Pitch Bend change status byte; nnnn (0-15) = MIDI channel 1-16	
vvvvvvv	Pitch Bend LSB value (0-127)	00 40 is the central (no bend) setting. 00 00 is the maximum downwards bend, and 7F 7F the maximum upwards bend.
hhhhhhh	Pitch Bend MSB value (0-127)	

Notes :

1. The use of a 14-bit value for pitch bend control has the double advantage of allowing fine adjustment and providing a useful range.
2. Pitch Bend sensitivity is set manually at the receiving device, or by use of the Registered Parameter Number (RPN) 00 00 controller message.

Controller Numbers

The controller number is the first data byte (0ccccccc) following a Controller Change status byte (Bn). The 128 available controller numbers are split into four groups :

0-63	High resolution continuous controllers (0-31 = MSB; 32-63 = LSB)
64-69	Switches
70-119	Low resolution continuous controllers
120-127	Channel Mode messages

Note that for switches, the second data byte (0vvvvvvv) is either 0 (Off) or 127 (On), and that for high and low resolution continuous controllers, the second data byte takes the range 0-127.

Page content last updated : Mon 26 Sep 2022

The high resolution continuous controllers are divided into MSB and LSB values, providing a maximum of 14-bit resolution. If only 7-bit resolution is needed for a specific controller, only the MSB is used - it is not necessary to send the LSB. If the full resolution is required, then the MSB should be sent first, followed by the LSB. If only the LSB has changed in value, the LSB may be sent without re-sending the MSB.

The controller numbers missing from the following list (3, 9, 14, 15, 20-31, 35, 41, 46, 47, 52-63, 85-87, 89, 90 and 102-119) are currently undefined.

High resolution continuous controllers (MSB)

0	Bank Select (Detail)
1	Modulation Wheel
2	Breath Controller
4	Foot Controller
5	Portamento Time
6	Data Entry (used with RPNs/NRPNs)
7	Channel Volume
8	Balance
10	Pan
11	Expression Controller
12	Effect Control 1
13	Effect Control 2
16	Gen Purpose Controller 1
17	Gen Purpose Controller 2
18	Gen Purpose Controller 3
19	Gen Purpose Controller 4

High resolution continuous controllers (LSB)

32	Bank Select
33	Modulation Wheel
34	Breath Controller
36	Foot Controller
37	Portamento Time
38	Data Entry
39	Channel Volume
40	Balance
42	Pan
43	Expression Controller
44	Effect Control 1
45	Effect Control 2
48	General Purpose Controller 1
49	General Purpose Controller 2
50	General Purpose Controller 3
51	General Purpose Controller 4

Switches

64	Sustain On/Off
65	Portamento On/Off
66	Sostenuto On/Off
67	Soft Pedal On/Off
68	Legato On/Off
69	Hold 2 On/Off

Page content last updated : Mon 26 Sep 2022

Low resolution continuous controllers

70 Sound Controller 1 (TG: Sound Variation; FX: Exciter On/Off)
71 Sound Controller 2 (TG: Harmonic Content; FX: Compressor On/Off)
72 Sound Controller 3 (TG: Release Time; FX: Distortion On/Off)
73 Sound Controller 4 (TG: Attack Time; FX: EQ On/Off)
74 Sound Controller 5 (TG: Brightness; FX: Expander On/Off)
75 Sound Controller 6 (TG: Decay Time; FX: Reverb On/Off)
76 Sound Controller 7 (TG: Vibrato Rate; FX: Delay On/Off)
77 Sound Controller 8 (TG: Vibrato Depth; FX: Pitch Transpose On/Off)
78 Sound Controller 9 (TG: Vibrato Delay; FX: Flange/Chorus On/Off)
79 Sound Controller 10 (TG: Undefined; FX: Special Effects On/Off)
80 General Purpose Controller 5
81 General Purpose Controller 6
82 General Purpose Controller 7
83 General Purpose Controller 8
84 Portamento Control (PTC) (`0vvvvvvv` is the source Note number) (Detail)
88 High Resolution Velocity Prefix
91 Effects 1 Depth (Reverb Send Level)
92 Effects 2 Depth (Tremelo Depth)
93 Effects 3 Depth (Chorus Send Level)
94 Effects 4 Depth (Celeste Depth)
95 Effects 5 Depth (Phaser Depth)

RPNs / NRPNs - (Detail)

96 Data Increment
97 Data Decrement
98 Non Registered Parameter Number (LSB)
99 Non Registered Parameter Number (MSB)
100 Registered Parameter Number (LSB)
101 Registered Parameter Number (MSB)

Channel Mode messages - (Detail)

120 All Sound Off
121 Reset All Controllers
122 Local Control On/Off
123 All Notes Off
124 Omni Mode Off (also causes ANO)
125 Omni Mode On (also causes ANO)
126 Mono Mode On (Poly Off; also causes ANO)
127 Poly Mode On (Mono Off; also causes ANO)

Bank Select

Controller numbers 0 (MSB) and 32 (LSB) are used for Bank selection.

The concept of instrument banks was introduced to get around the 128 voice limit of the **Program Change** message. The bank number is a 14-bit value, hence 16384 different banks are theoretically possible. The **Program Change** message gives access to up to 128 voices, from within the currently selected bank. There is, however, no requirement for a bank to contain the full set of 128 voices.

The **Bank Select** message should be transmitted as a pair, and then followed by a **Program Change**

Page content last updated: Mon 26 Sep 2022

message :

Hex	Binary	Description
Bn , 00 , msb	1011nnnn , 00000000 , 0vvvvvvv	Bank Select, MSB
Bn , 20 , lsb	1011nnnn , 00100000 , 0vvvvvvv	Bank Select, LSB
Cn , pc	1100nnnn , 0ppppppp	Program Change

The 14-bit Bank Select value gives access to 16,384 banks using the formula :

(MSB * 128) + LSB + 1

E.g. :

MSB	LSB	Bank number
00	00	1
00	7F	128
01	00	129
7F	7F	16,384

Portamento Control (PTC)

Controller number 84 is used for **Portamento Control**.

When a Note On message is received after a PTC message, the voice's pitch glides from the key number specified in the PTC message to the new Note On's pitch at the rate set by Controller 5 (**Portamento Time**), ignoring the current status of Controller 65 (**Portamento On/Off**).

A PTC message only affects the next Note On received on the relevant MIDI channel.

When in **Poly** mode, receiving a PTC message does not affect the pitch of any currently playing notes (whether in their sustain or release phase).

When in **Mono** mode, or if Controller 68 (**Legato footswitch**) is On, a new overlapping note event results in an immediate pitch jump to the note number specified in the PTC message, and then a glide at the current portamento rate to the note number specified in the new Note On.

For an explanation of Mono and Poly modes see the MIDI Modes sub-section under Channel Mode Messages.

The note is turned off by a Note Off which matches the Note On key number, not the key number specified in the PTC message, irrespective of mode.

Example usage :

To perform a glide from C4 to E4 :

MIDI data	Description	Result
90 3C 40	Note On, C4	C4 On
B0 54 3C	PTC from C4	No audible change
90 40 40	Note On, E4	Glide from C4 to E4
80 3C 40	Note Off, C4	No audible change
80 40 40	Note Off, E4	E4 Off

Page content last updated : Mon 26 Sep 2022

RPNs and NRPNs

Controller numbers 98 & 99 (**Non-Registered Parameter Number**, LSB & MSB), and 100 & 101 (**Registered Parameter Number**, LSB & MSB), in conjunction with Controller numbers 6 & 38 (**Data Entry**, MSB & LSB), 96 (**Data Increment**), and 97 (**Data Decrement**) extend the number of controllers available via MIDI.

Their use involves selecting the parameter number to be edited using Controllers 98 & 99 or 100 & 101, and then adjusting the value for that parameter using Controller number 6/38, 96, or 97. Controllers 6/38 would be used to set a specific value, whereas Controllers 96 and 97 would be used to nudge the current value up or down, respectively (with the Data2 byte specifying the step size, 1-127).

Note that two 7-bit values are used both to specify the RPN/NRPN itself, and its value. I.e. there are 16,384 possible RPNs (and 16,384 NRPNs) each of which can be set with 14-bit precision. The MSB and LSB data values need not represent a single 0-16,383 range, and can be used to represent different quantities, as in the case of RPN 00,00 (**Pitch Bend Sensitivity**) where the MSB represents 'semitones' and the LSB represents 'cents'.

To calculate the full number from its LSB and MSB : N = MSB * 128 + LSB

Once the required RPN/NRPN has been specified (using CC98 & CC99, or CC100 & CC101) the value needs to be specified. There are several options :

- High resolution values (i.e. ranging 0-16,383) require two messages :
  - A CC6 **Data Entry** message, where the Data2 byte contains the MSB of the value.
  - A CC38 **Data Entry** message, where the Data2 byte contains the LSB of the value.
- Low resolution values (i.e. 0-127) can be sent in one message. It could be either a CC6 or a CC38 **Data Entry** message.
- A CC96 **Data Increment** message can be sent. Its Data2 byte holds a value (0-127) to be added to the current parameter value.
- A CC97 **Data Decrement** message can be sent. Its Data2 byte holds a value (0-127) to be subtracted from the current parameter value.

None of the **Non-Registered Parameter Numbers** have been assigned specific functions. They may be used for different functions by different manufacturers, and are thus manufacturer-specific.

Registered Parameter Numbers are those which have been assigned some particular function by the MIDI Manufacturers Association (MMA) and the Japan MIDI Standards Committee (JMSC). The following RPNs are currently defined :

RPN		Description	
MSB	LSB		
00	00	**Pitch Bend Sensitivity** The coarse adjustment (Controller 6) sets the number of semitones. The fine adjustment (Controller 38) sets the number of cents.	
00	01	**Channel Fine Tuning** Use controllers 6 and 38 to set MSB and LSB : 00 00 = -100 cents; 40 00 = A440; 7F 7F = +100 cents.	See also the **Real Time Universal System Exclusive** Master Tuning messages.
00	02	**Channel Coarse Tuning** Coarse adjustment only, using controller 6 : 00 = -64 semitones; 40 = A440; 7F = +63 semitones.	

Page content last updated Mon 26 Sep 2022

00	03	**Select Tuning Program**	See also the **Real Time Universal System Exclusive** MIDI Tuning Standard messages.
00	04	**Select Tuning Bank**	
00	05	**Modulation Depth Range** Used to scale the effective range of Controller 1 (Modulation Wheel).	
7F	7F	**Null Function** Used to cancel a RPN or NRPN. After it is received, any additional value updates received should no longer be applied to the previously selected RPN or NRPN.	

It is recommended that the **Null Function** (RPN 7F,7F) should be sent immediately after a RPN or NRPN and its value are sent.

Example usage :

To set the Pitch Bend Sensitivity on channel 'n' to +/- 7 semitones (ie +/- a fifth) :

11 bytes	`Bn 64 00 65 00 06 07 64 7F 65 7F`	
Bn 64 00	RPN LSB = 00	Select RPN : Pitch Bend Sensitivity
65 00	RPN MSB = 00 (running status in effect)	
06 07	Data Entry MSB = 07 (running status in effect)	Coarse adjustment (semitones)
64 7F	RPN LSB = 7F (running status in effect)	Null Function (Cancel RPN)
65 7F	RPN MSB = 7F (running status in effect)	

Or, to set Tuning Program 'tt' on channel 'n' :

11 bytes	`Bn 64 03 65 00 06 tt 64 7F 65 7F`	
Bn 64 03	RPN LSB = 03	Select RPN : Select Tuning Program
65 00	RPN MSB = 00 (running status in effect)	
06 tt	Data Entry MSB = tt (running status in effect)	Coarse adjustment
64 7F	RPN LSB = 7F (running status in effect)	Null Function (Cancel RPN)
65 7F	RPN MSB = 7F (running status in effect)	

Or, to increment (by 1) the current Tuning Bank on channel 'n' :

11 bytes	`Bn 64 04 65 00 60 01 64 7F 65 7F`	
Bn 64 04	RPN LSB = 04	Select RPN : Select Tuning Bank
65 00	RPN MSB = 00 (running status in effect)	
60 01	Data Increment (running status in effect)	Increment (by 1)
64 7F	RPN LSB = 7F (running status in effect)	Null Function (Cancel RPN)
65 7F	RPN MSB = 7F (running status in effect)	

When sending successive RPN (or NRPN) messages, the standard allows the ommision of CC100 & CC101 (or CC98 & CC99) if their value is unchanged.

For example, to send a pair of **Channel Coarse Tuning** and **Channel Fine Tuning** RPN messages (with a low-res and high-res data value, respectively) :

17 bytes	`Bn 64 02 65 00 06 TM 64 01 06 tM 26 tL 64 7F 65 7F`	
Bn 64 02	RPN LSB = 02	Select RPN : Channel Coarse Tuning
65 00	RPN MSB = 00 (running status in effect)	
06 TM	Data Entry, MSB (running status in effect)	Coarse adjustment (of 'coarse tuning')
64 01	RPN LSB = 01 (running status in effect)	Select RPN : Channel Fine Tuning (RPN MSB = 00 is still in effect)

Page content last updated : Mon 26 Sep 2022

06 tM	Data Entry, MSB (running status in effect)	Coarse adjustment (of 'fine tuning')
26 tL	Data Entry, LSB (running status in effect)	Fine adjustment (of 'fine tuning')
64 7F	RPN LSB = 7F (running status in effect)	Null Function (Cancel RPN)
65 7F	RPN MSB = 7F (running status in effect)	

Channel Mode Messages

These are a subset of the Controller messages, specifically controller numbers 120 - 127.

3 bytes	`1011nnnn , 0ccccccc , 0vvvvvvv`

ccccccc	vvvvvvv	Description
120	0	**All Sound Off** Causes sound generators to immediately cease all sound (no *release* phase involved).
121	0	**Reset All Controllers** Resets all controllers to their default values.
122	0 (Off) or 127 (On)	**Local Control On/Off** When Local Control is Off, MIDI data from a MIDI keyboard is not communicated to the keyboard's sound generators, though it is still sent out via the MIDI OUT port. MIDI data arriving at the MIDI IN port will reach the sound generators, regardless.
123	0	**All Notes Off (ANO)** Causes sound generators to enter the *release* stage of their volume envelope, and hence *gently* cease making sound.
124	0	**Omni Mode Off** (also causes **ANO**)
125	0	**Omni Mode On** (also causes **ANO**)
126	0-16	**Mono Mode On (Poly Off)** (also causes **ANO**) vvvvvvv > 0 : Number of channels to use (Omni Off). vvvvvvv = 0 : Use all available channels (Omni On)
127	0	**Poly Mode On (Mono Off)** (also causes **ANO**)

MIDI Modes

The MIDI specification allows for four basic modes of operation, selected using any two from the last four Channel Mode messages from the above list.

When **Omni Mode** is on, a device will respond to Channel messages on any MIDI channel, whereas when **Omni Mode** is off, a device will only respond to Channel messages which are on that device's *base channel*.

When **Mono Mode** is on, a device will only be able to play a single note at any time on a given MIDI channel, whereas when **Poly Mode** is on, a device will be able to play chords on a given channel (within that device's polyphony capability).

Note that any of these Channel Mode messages will only be responded to if they are on the receiving device's base channel. So, even when in **Omni** mode, a device still has a base channel as such (for the purposes of these Channel Mode messages).

Omni Mode On is a failsafe approach when connecting two devices together, as it ensures that the

Page content last updated : Mon 26 Sep 2022

receiving device will respond to messages sent by the transmitting device. It is, however, totally useless in a sequencer based setup.

Multitimbral devices operate in a slightly special way. Basically each *part* (which has its own MIDI channel) is considered as a separate device, with each part being set to **Omni Mode Off** and **Poly Mode On** (i.e. independent polyphonic channels), which is ideal for use in a sequencer setup.

1 : Omni On, Poly On

Bn, 7D, 00	Omni Mode On
Bn, 7F, 00	Poly Mode On

Voice messages are received from *all* voice channels and assigned to voices polyphonically.

All voices are transmitted on the *base channel*.

2 : Omni On, Mono On

Bn, 7D, 00	Omni Mode On
Bn, 7E, 00	Mono Mode On

Voice messages are received from *all* voice channels, and control only one voice, monophonically.

Voice messages for one voice are transmitted on the *base channel*.

3 : Omni Off, Poly On

Bn, 7C, 00	Omni Mode Off
Bn, 7F, 00	Poly Mode On

Voice messages are received on the *base channel* only, and assigned to voices polyphonically.

Voice messages for *all* voices are transmitted on the *base channel*.

4 : Omni Off, Mono On

Bn, 7C, 00	Omni Mode Off
Bn, 7E, nn	Mono Mode On (nn = number of channels to use)

Voice messages are received on voice channels *<base channel>* to *<base channel> + nn - 1*, and assigned monophonically to voices 1 to nn, respectively. The number of voices 'nn' is specified by the third byte of the **Mono Mode** message.

Voice messages for voices 1 to nn are transmitted on voice channels *<base channel>* to *<base channel> + nn - 1*, respectively (a single voice per channel).

System Common Messages

System Common information is intended for all channels in a system.

Status byte		Data bytes	Description
Hex	Binary		

Page content last updated : Mon 26 Sep 2022

F1	11110001	0nnndddd	—	**MIDI Time Code Quarter Frame** nnn = Message Type; dddd = values. See also the **Real Time Universal System Exclusive** MTC and Bar Marker (Notation) messages.
F2	11110010	0lllllll	0hhhhhhh	**Song Position Pointer** lllllll = LSB (0-127); hhhhhhh = MSB (0-127) The 14-bit value is the number of MIDI beats since the start of the song (1 beat = 6 MIDI clocks).
F3	11110011	0sssssss	—	**Song Select** sssssss = Song number (0-127) Select current sequence.
F4	11110100			**Undefined**
F5	11110101			**Undefined**
F6	11110110	—	—	**Tune Request** Upon receiving this all analogue synthesizers should initiate a self tuning operation.
F7	11110111	—	—	**End of System Exclusive** (EOX) Used to terminate a System Exclusive message.

MTC Quarter Frame

These are the MTC (i.e. SMPTE based) equivalent of the F8 Timing Clock messages, though offer much higher resolution, as they are sent at a rate of 96 to 120 times a second (depending on the SMPTE frame rate). Each **Quarter Frame** message provides partial timecode information, 8 sequential messages being required to fully describe a timecode instant. The reconstituted timecode refers to when the first partial was received. The most significant nibble of the data byte indicates the partial (aka Message Type).

Partial	Data byte	Usage	
1	0000 bcde	Frame number LSBs	abcde = **Frame number** (0 to frameRate-1)
2	0001 000a	Frame number MSB	
3	0010 cdef	Seconds LSBs	abcdef = **Seconds** (0-59)
4	0011 00ab	Seconds MSBs	
5	0100 cdef	Minutes LSBs	abcdef = **Minutes** (0-59)
6	0101 00ab	Minutes MSBs	
7	0110 defg	Hours LSBs	ab = **Frame Rate** (00 = 24, 01 = 25, 10 = 30drop, 11 = 30nondrop) cdefg = **Hours** (0-23)
8	0111 0abc	Frame Rate, and Hours MSB	

System Real Time Messages

The **System Real Time** messages control the entire system (all devices, irrespective of channel setting) in real time. They are used for synchronising clock-based devices (e.g. sequencers and rhythm units). They are each single byte messages, with no following data bytes. If the functions specified are not implemented, they are simply ignored.

To maintain timing precision, these messages can be sent at any time, even between bytes of another message. In such a situation the Real Time message is either acted upon or ignored, after which the receiving process resumes under the previous status.

Status byte		Description
Hex	Binary	

Page content last updated : Mon 26 Sep 2022

F8	11111000	**Timing Clock** Whilst a transmitter is *playing*, this message is sent 24 times per quarter note.
F9	11111001	**Undefined**
FA	11111010	**Start** Start the current sequence playing, from the beginning. (This message should be followed with **Timing Clocks**).
FB	11111011	**Continue** Continue playing the current sequence from the point at which it was stopped.
FC	11111100	**Stop** Stop the current sequence.
FD	11111101	**Undefined**
FE	11111110	**Active Sensing** When initially sent, the receiver will expect to receive another Active Sensing message each 300ms (max), or it will assume that the connection has been terminated. At termination, the receiver will turn off all voices and return to normal (non- active sensing) operation. Use of this message is optional.
FF	11111111	**System Reset** Reset all receivers to their power-up status. It should be used sparingly, preferably only under manual control. In particular, it should not be sent automatically on power up, as this could lead to a situation where two devices endlessly reset each other.

System Exclusive Messages

Status byte		Data bytes	Description
Hex	Binary		
F0	11110000		**Start of System Exclusive** (SOX)
		0iiiiiii	Identification code (0-127) [see notes 1 & 5]
		0ddddddd 0ddddddd	Any number of data bytes (each 0-127) having manufacturer specific functionality
F7	11110111		**End of System Exclusive** (EOX)

Notes :

1. iiiiiii = identification ID (0-127). If the ID is 0, the following 2 bytes are used as extensions to the manufacturer ID.
2. All bytes between the System Exclusive status byte and EOX, or the next status byte, must have zero in the top bit, i.e. values 0-127.
3. Status or Data bytes (except System Real Time) should not be interleaved with System Exclusive data.
4. EOX, or any other status byte (except System Real Time) will terminate a System Exclusive message. However, an EOX should always be sent to specifically mark the end of a System Exclusive message.
5. 3 System Exclusive ID numbers have been set aside for special purposes : 7D is reserved for non-commercial use (e.g. schools, research, etc.) and is not to be used on any product released to the public; 7E (Non Real Time) and 7F (Real Time) are used for extensions to the MIDI specification, and are known as Universal System Exclusive ID numbers.

System Exclusive Manufacturer's ID Numbers

This is by no means a complete list, though it does include the major companies.

See note 1 above for how to distinguish single and multi byte IDs.

Page content last updated : Mon 26 Sep 2022

Single byte IDs - American

01	Sequential Circuits
04	Moog
05	Passport Designs
06	Lexicon
07	Kurzweil
08	Fender
0A	AKG Acoustics
0F	Ensoniq
10	Oberheim
11	Apple
13	Digidesign
15	JL Cooper
18	Emu
1A	ART
1C	Eventide

Single byte IDs - European

22	Synthaxe
24	Hohner
29	PPG
2B	SSL
2D	Neve
2F	Elka / General Music
30	Dynacord
33	Clavia (Nord)
36	Cheetah
3A	Steinberg Media Technologies GmbH
3E	Waldorf Electronics Gmbh

Single byte IDs - Japanese

40	Kawai
41	Roland
42	Korg
43	Yamaha
44	Casio
47	Akai
48	Japan Victor (JVC)
4C	Sony
4E	Teac Corporation
51	Fostex
52	Zoom

Multi byte IDs - American

00 00 07	Digital Music Corporation
00 00 09	New England Digital
00 00 0E	Alesis
00 00 15	KAT Inc
00 00 16	Opcode
00 00 1A	Allen & Heath Brenell
00 00 1B	Peavey Electronics
00 00 1C	360 Systems

Page content last updated : Mon 26 Sep 2022

00 00 1F	Zeta Systems
00 00 20	Axxes
00 00 3B	Mark Of The Unicorn (MOTU)
00 00 41	Microsoft
00 00 4D	Studio Electronics
00 00 50	MIDI Solutions Inc
00 00 66	Mackie
00 01 05	M-Audio (Midiman)
00 01 06	PreSonus
00 01 36	Radikal Technologies
00 01 37	Roger Linn Design
00 01 5A	Plogue Art et Technologie
00 01 61	Livid
00 01 6C	Source Audio LLC
00 01 72	Kilpatrick Audio
00 01 73	iConnectivity
00 02 10	Meris LLC
00 02 14	Intellijel Designs Inc
00 02 1D	Sensel Inc
00 02 26	Electro-Harmonix
00 02 2C	Spectrasonics Inc
00 02 44	Noise Engineering

Multi byte IDs - European

00 20 11	Forefront Technology
00 20 13	Kenton Electronics
00 20 1F	TC Electronic
00 20 20	Doepfer
00 20 27	Acorn Computer
00 20 29	Focusrite / Novation
00 20 32	Behringer
00 20 33	Access Music Electronics
00 20 3C	Elektron
00 20 47	Klavis Technologies
00 20 4D	Vermona
00 20 52	Analogue Systems
00 20 64	genoQs Machines GmbH
00 20 69	Elby Designs
00 20 6B	Arturia
00 20 6D	C-Thru Music
00 20 70	OTO Machines
00 20 76	Teenage Engineering
00 21 02	Mutable Instruments
00 21 03	PreSonus Software Ltd
00 21 07	Modal Electronics
00 21 09	Native Instruments
00 21 10	ROLI Ltd
00 21 17	Rob Papen
00 21 1A	IK Multimedia
00 21 1C	Modor Music
00 21 1D	Ableton

Page content last updated : Mon 26 Sep 2022

00 21 23 Retrokits
00 21 26 Expressive E
00 21 27 Expert Sleepers
00 21 2A Sonic Potions
00 21 32 Bome Sftware
00 21 35 Dreadbox P.C.
00 21 38 ALM Co (Busy Circuits)
00 21 3B Blokas
00 21 49 Bitwig GmbH

Multi byte IDs - Japanese

00 40 06 Pioneer Corporation

Universal System Exclusive ID Numbers

This includes the **Non Real Time** (7E) and **Real Time** (7F) ID extensions mentioned above in note 5.

The generalised format for both is as follows :

F0 <ID number> <Device ID> <Sub ID#1> <Sub ID#2> F7

The **Device ID** (also referred to as the **channel number**) is generally used to specify a discrete physical device, however complex devices (e.g. computers with a number of different MIDI expansion cards) may have more than one Device ID. A value of 7F is used to specify *all devices*.

From one to sixteen virtual devices may be accessed at each Device ID by use of the normal MIDI channel numbers, within the capabilities of the device.

The **Non-Commercial** Universal System Exclusive ID (7D) is not detailed here.

Sub ID#1	Sub ID#2	Description
Non Real Time Universal Sys Ex (Sys Ex ID number = 7E)		
00	—	**Unused**
01	(not used)	**Sample Dump Header** - (Detail)
02	(not used)	**Sample Dump Data Packet** - (Detail)
03	(not used)	**Sample Dump Request** - (Detail)
04	nn	**MTC Cueing** - (Detail)
	00	Special
	01	Punch In Points
	02	Punch Out Points
	03	Delete Punch In Point
	04	Delete Punch Out Point
	05	Event Start Point
	06	Event Stop Point
	07	Event Start Points with additional info
	08	Event Stop Points with additional info
	09	Delete Event Start Point
	0A	Delete Event Stop Point
	0B	Cue Points
	0C	Cue Points with additional info
	0D	Delete Cue Point

	0E	Event name in additional info
05	nn	**Sample Dump Extensions**
	01	Loop Point Transmission - (Detail)
	02	Loop Point Request - (Detail)
	03	Sample Name Transmission - (Detail)
	04	Sample Name Request - (Detail)
	05	Extended Dump Header - (Detail)
	06	Extended Loop Point Transmission - (Detail)
	07	Extended Loop Point Request - (Detail)
06	nn	**General System Information**
	01	Device Identity Request - (Detail)
	02	Device Identity Reply - (Detail)
07	nn	**File Dump**
	01	Header - (Detail)
	02	Data Packet - (Detail)
	03	Request - (Detail)
08	nn	**MIDI Tuning Standard**
	00	Bulk Tuning Dump Request - (Detail)
	01	Bulk Tuning Dump Reply - (Detail)
09	nn	**General MIDI (GM) System**
	01	Enable - (Detail)
	02	Disable - (Detail)
0A	nn	**Down-Loadable Sounds (DLS) System**
	01	Enable - (Detail)
	02	Disable - (Detail)
7B	(not used)	**End of File** - (Detail)
7C	(not used)	**Wait** - (Detail)
7D	(not used)	**Cancel** - (Detail)
7E	(not used)	**NAK** - (Detail)
7F	(not used)	**ACK** - (Detail)
Real Time Universal Sys Ex (Sys Ex ID number = 7F)		
00	—	**Unused**
01	nn	**MIDI Time Code (MTC)**
	01	Full Message - (Detail)
	02	User Bits - (Detail)
03	nn	**Notation Information**
	01	Bar Marker - (Detail)
	02	Time Signature (immediate) - (Detail)
	42	Time Signature (delayed) - (Detail)
04	nn	**Device Control**
	01	Master Volume - (Detail)
	02	Master Balance - (Detail)
	03	Master Fine Tuning - (Detail)
	04	Master Coarse Tuning - (Detail)
05	nn	**MTC Cueing** - (Detail)
	00	Special
	01	Punch In Points
	02	Punch Out Points
	05	Event Start Point

Page content last updated : Mon 26 Sep 2022

	06	Event Stop Point
	07	Event Start Points with additional info
	08	Event Stop Points with additional info
	0B	Cue Points
	0C	Cue Points with additional info
	0E	Event name in additional info
06	nn	**MIDI Machine Control (MMC) Commands** - (Detail)
	01	Stop
	02	Play
	03	Deferred Play
	04	Fast Forward
	05	Rewind
	06	Record Strobe (Punch In)
	07	Record Exit (Punch Out)
	08	Record Pause
	09	Pause
	0A	Eject
	0B	Chase
	0C	Command Error Reset
	0D	MMC Reset
	44	Locate / Go To - (Detail)
	47	Shuttle - (Detail)
08	nn	**MIDI Tuning Standard**
	02	Single Note Tuning Change - (Detail)

Non Real Time Universal System Exclusive Messages

Sample Dump Standard (SDS)

This standard enables the transfer of samples over MIDI. It has been designed to work either as an open or closed loop system. The closed loop system involves handshaking procedures which allows for error recovery, and also accommodates devices which need time to process the data as it is received.

The three SDS messages (**Dump Request**, **Dump Header** and **Data Packet**) are used in combination with the five Handshaking messages (**EOF**, **ACK**, **NAK**, **Wait** and **Cancel**) in accordance with the Dump Procedure.

See also the Sample Dump Extensions messages.

There is also a corresponding set of File Dump messages.

Sample Dump Header

F0 7E id 01 ss ss ee ff ff ff gg gg gg hh hh hh ii ii ii jj F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (7F = All devices)
01	Sub ID#1 = Sample Dump Header

Page content last updated : Mon 26 Sep 2022

ss ss	Sample number (LSB first)
ee	Sample format (number of significant bits : 8-28)
ff ff ff	Sample period (1/sample rate) in nanoseconds (LSB first)
gg gg gg	Sample length in words (LSB first)
hh hh hh	Sustain loop start point (word number, LSB first)
ii ii ii	Sustain loop end point (word number, LSB first)
jj	Loop type (00 = forward only, 01 = backward/forward, 7F = Off)
F7	EOX

Sample Dump Data Packet

F0 7E id 02 nn <120 bytes> kk F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (7F = All devices)
02	Sub ID#1 = Sample Dump Data Packet
nn	Running packet count (0-127, with wrap-around)
<120 bytes>	Sample data (each byte contains 7-bit data)
kk	Checksum (XOR of : 7E cc 02 nn <120 bytes>)
F7	EOX

The packet number (nn) is used by the receiver to distinguish between a re-send and a new packet, when operating in handshaking mode. Note that for sample sizes that require more than 128 data packets, the packet number wraps around (from 127 to 0).

The data is in the usual 7-bit format, i.e. with the msb = 0. The number of MIDI bytes used to encode each *sample* depends on the sample format :

- 8-14 bit samples : 2 bytes per sample (60 samples per packet).
- 15-21 bit samples : 3 bytes per sample (40 samples per packet).
- 22-28 bit samples : 4 bytes per sample (30 samples per packet).

Data is encoded left justified within the 7-bit bytes, with any unused (rightmost) bits being set to zero.

An example to clarify that :

The 16-bit sample : `5B3E (0101 1011 0011 1110)`

would be encoded as : `00101101 01001111 01000000`

Sample Dump Request

Upon receiving this message, a sampler should check to see if the requested sample number is valid (i.e. that it is within range). If so, then the requested sample is dumped, a packet at a time to the requesting device (using the handshaking messages as described in the Dump Procedure).

F0 7E id 03 ss ss F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (7F = All devices)
03	Sub ID#1 = Sample Dump Request
ss ss	Sample number (LSB first)
F7	EOX

Back to index of Non Real Time Universal Sys Ex IDs

Page content last updated : Mon 26 Sep 2022

Handshaking

There are five handshaking messages (**EOF**, **ACK**, **NAK**, **Wait** and **Cancel**), used in association with the Sample Dump Standard or File Dump messages. Their use is described in the Dump Procedure.

End of File (EOF)

This message signifies that the end of the file has been reached and that the data transfer has been completed successfully.

F0 7E id 7B pp F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (7F = All devices)
7B	Sub ID#1 = End of File (EOF)
pp	Packet number
F7	EOX

Wait

The use of this message implies *'Do not send any more packets until told to do so'*. This is important for systems where the receiver needs time to perform other processing before receiving the remainder of the sample dump. An **ACK** will resume the dump, whereas a **Cancel** will abort it.

F0 7E id 7C pp F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (7F = All devices)
7C	Sub ID#1 = Wait
pp	Packet number
F7	EOX

Cancel

The use of this message implies *'Abort dump'*. The packet number refers to the packet on which the abort takes place.

F0 7E id 7D pp F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (7F = All devices)
7D	Sub ID#1 = Cancel
pp	Packet number
F7	EOX

NAK

The use of this message implies *'The last data packet was received incorrectly - Please re-send'*. The packet number refers to the packet being rejected.

F0 7E id 7E pp F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (7F = All devices)
7E	Sub ID#1 = NAK

Page content last updated : Mon 26 Sep 2022

pp	Packet number
F7	EOX

ACK

The use of this message implies *'The last data packet was received correctly - Start sending the next one'*. The packet number refers to the packet being acknowledged as correct.

F0 7E id 7F pp F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (7F = All devices)
7F	Sub ID#1 = ACK
pp	Packet number
F7	EOX

Back to index of Non Real Time Universal Sys Ex IDs

Dump Procedure

This applies to the Sample Dump and File Dump standards and involves the transfer of data from **sender** to **receiver**. The transfer process can be initiated by either device.

The receiver can initiate the process by sending the appropriate (Sample or File) **Dump Request** message to the sender, whereas the sender can initiate the process by sending the appropriate (Sample or File) **Dump Header** message.

Sender

Either upon reception of a **Dump Request**, or in response to user input on the device's front panel, a **Dump Header** message is sent.

A *timeout* scheme is used to determine whether the open loop (no handshaking) or closed loop (with handshaking) protocol is to be used. The sender will wait for up to 2 seconds for a response, after which it will time out. Within this timeout period :

- If it receives a **Cancel**, it will immediately abort the dump process.
- If it receives an **ACK**, it will begin transfer by sending the first **Data Packet**.
- If it receives a **Wait**, it will suspend transfer indefinitely until another message (**ACK** or **Cancel**) is received, which will then be processed as above.
- If nothing is received within the timeout period (2s), the open loop protocol is assumed, and the dump starts with the first packet.

After sending each packet, the sender will wait at least 20ms for a response, after which it will time out. Within this timeout period :

- If it receives an **ACK**, it will send the next **Data Packet**.
- If it receives an **NAK**, and the packet number matches the number of the last packet sent, it will re-send that packet. If the packet numbers do not match, and the device is incapable of sending packets out of sequence, the **NAK** will be ignored, and the transfer process will continue.
- If it receives a **Wait**, it will suspend transfer indefinitely until one of **ACK**, **NAK**, or **Cancel** is received, whereupon it will continue.
- If no messages are received within the timeout period (20ms), the sender will assume that the

open loop protocol is in effect, and the next packet will be sent.

This process continues until there are less than 121 data bytes remaining to be sent. The final packet will still consist of 120 bytes, regardless, with the excess bytes all being 0.

Receiver

Whilst receiving a **Sample** or **File Data Packet**, a device should keep a running checksum. This calculated checksum can be compared with the checksum within the **Data Packet** to ascertain whether or not the data was received correctly :

- If the checksums match, the receiver will send an **ACK** and await the next packet.
- If the checksums do not match, it will send a **NAK** containing the number of the packet that caused the error, and await the next packet. If, after sending a **NAK**, the packet number of the next packet doesn't match the previous packet number (the one that was **NAK**'d), and the unit is not capable of accepting packets out of sequence, the error is ignored and the dump continues as if the checksums had matched.

If a receiver runs out of memory before the dump is completed, it will send a **Cancel** to stop the dump.

Back to index of Non Real Time Universal Sys Ex IDs

MTC Cueing

These messages allow timecode based event lists to be set up.

There is a corresponding set of Real Time MIDI Cueing messages, which take effect immediately.

Cueing Setup

F0 7E id 04 tt hr mn sc fr ff ee ee <info> F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
04	Sub ID#1 = MTC Cueing messages
tt	Sub ID#2 = Setup Type (see the following table)
hr	Hours and Type : 0yyzzzzz yy = Type; 00 = 24 fps, 01 = 25 fps, 10 = 30 fps (drop frame), 11 = 30 fps (non-drop frame). zzzzz = Hours (0-23).
mn	Minutes (0-59)
sc	Seconds (0-59)
fr	Frames (0-29)
ff	Fractional frames (0-99)
ee ee	Event number (LSB first)
<info>	Only present for Setup Types (tt) : 07, 08, 0C and 0E (see note 1 below)
F7	EOX

tt	Description

Page content last updated : Mon 26 Sep 2022

<table>
<tr><td>00</td><td colspan="2">Special is used to setup a unit as a whole (as opposed to individual sequences, sounds, etc.) Note that each of the following event types, except Event List Request ignore the event time field.

ee ee — Description
00 00 — Time Code Offset specifies a relative offset for the receiving device.
01 00 — Enable Event List instructs the receiving device to execute events in its event list at the appropriate time.
02 00 — Disable Event List instructs the receiving device to not execute events in its event list. Essentially it acts as a mute control.
03 00 — Clear Event List instructs the receiving device to erase its entire event list.
04 00 — System Stop specifies the time at which a unit should shut down. Used to (eg) prevent a tape recorder running past the end of the reel.
05 00 — Event List Request is used to request a list of Cueing Setup messages which describe the receiving device's event list, starting from the SMPTE time specified in the request message.</td></tr>
<tr><td>01</td><td>Punch In</td><td rowspan="2">These enable / disable record mode for the receiving device. The Event Number (ee ee) specifies the track number. Multiple points may be specified by sending a series of Cueing Setup messages with different times.</td></tr>
<tr><td>02</td><td>Punch Out</td></tr>
<tr><td>03</td><td>Delete Punch In</td><td rowspan="2">These delete the matching point, i.e. Time and Track (Event Number), from the cue list.</td></tr>
<tr><td>04</td><td>Delete Punch Out</td></tr>
<tr><td>05</td><td>Event Start</td><td rowspan="2">These refer to the running or playback of an event, and are used when a long sequence of events or a continuous event is to be started / stopped. Hence an event is a macro. The Event Number (ee ee) specifies which event is being refered to. A single event (e.g. mixer fader movement, playback of a sample, etc) may occur several times throughout a Cue List. These events will be refered to by the same event number, with different Start and Stop times.</td></tr>
<tr><td>06</td><td>Event Stop</td></tr>
<tr><td>07</td><td>Event Start + Info</td><td rowspan="2">As above, though with additional information (see note 1, below).</td></tr>
<tr><td>08</td><td>Event Stop + Info</td></tr>
<tr><td>09</td><td>Delete Event Start</td><td rowspan="2">These delete the matching event (Time and Event Number) whether with or without additional information, from the Cue List.</td></tr>
<tr><td>0A</td><td>Delete Event Stop</td></tr>
<tr><td>0B</td><td>Cue Point</td><td>This is a marker for a particular occurrence, e.g. an editing reference, or sound effects point. Each Cue Number is assigned to such an occurrence. (Use Event Start/Stop for continuous events.)</td></tr>
<tr><td>0C</td><td>Cue Point + Info</td><td>This is like Event Start/Stop with additional Info, except that the event represents a Cue Point rather than a Start/Stop Point. See note 1 below.</td></tr>
<tr><td>0D</td><td>Delete Cue Point</td><td>Deletes the matching Cue Event (Time and Event Number) whether with or without additional information, from the Cue List.</td></tr>
<tr><td>0E</td><td>Event Name in Info</td><td>This is used to label an Event Number. See note 1 below.</td></tr>
</table>

Notes :

1. Additional Information is a stream of nibblised MIDI data, LS nibble first, except when Setup Type = 0E, where it is nibblised ASCII data, again LS nibble first. An ASCII newline comprises a CR followed by a LF.

Back to index of Non Real Time Universal Sys Ex IDs

Sample Dump Extensions

With the passage of time since the original Sample Dump Standard, a number of shortcomings had become apparent :

- Only a single loop could be specified.
- An inability to name samples.

- A 2MB sample limit.
- Limited precision for describing the sample rate.

The following messages were added as an extension to the Sample Dump Standard, to address these limitations.

The first two messages overcome the 'single loop' shortcoming by allowing up to 16,383 pairs of loop points to be defined for each sample. They also allow loop points to be modified without having to re-send the entire sample.

Loop Point Transmission

F0 7E id 05 01 ss ss nn nn dd bb bb bb ee ee ee F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
05	Sub ID#1 = Sample Dump Extensions
01	Sub ID#2 = Loop Point Transmission
ss ss	Sample number (LSB first)
nn nn	Loop number (LSB first): 00 00 = the Sustain loop; 7F 7F = Delete all loops
dd	Loop type: 00 = Forwards only (unidirectional); 01 = Backwards/Forwards (bi-directional); 7F = Off
bb bb bb	Loop start address (in samples; LSB first)
ee ee ee	Loop end address (in samples; LSB first)
F7	EOX

Loop Point Request

F0 7E id 05 02 ss ss nn nn F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
05	Sub ID#1 = Sample Dump Extensions
02	Sub ID#2 = Loop Point Request
ss ss	Sample number (LSB first)
nn nn	Loop number (LSB first): 7F 7F = Request all loops
F7	EOX

With the exception of the **Request all loops** and **Delete all loops** commands (when Loop Number = `7F 7F`), the **Loop Point Request** and **Loop Point Transmission** messages affect a single loop point (start and end points).

Loop number `00 00` is the same as the sustain loop as defined in the SDS.

The following two messages provide support for sample names.

Sample Name Transmission

F0 7E id 05 03 ss ss tt <tag> nn <name> F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
05	Sub ID#1 = Sample Dump Extensions
03	Sub ID#2 = Sample Name Transmission
ss ss	Sample number (LSB first)

Page content last updated : Mon 26 Sep 2022

tt	Sample Name Language Tag Length (default = 00)
<tag>	Sample Language Tag Data (data comprising tt bytes)
nn	Sample Name Length (up to 127 characters)
<name>	Sample Name Data (string of length nn bytes)
F7	EOX

Note
The Sample Name Language Tag allows for future expansion regarding support for other languages. The default setting is zero, which defaults to English, and in this case the Name Data field is encoded using standard ASCII.

Sample Name Request

F0 7E id 05 04 ss ss F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
05	Sub ID#1 = Sample Dump Extensions
04	Sub ID#2 = Sample Name Request
ss ss	Sample number (LSB first)
F7	EOX

Extended Dump Header

This message gets around the 2MB sample size limit imposed by the SDS, whilst also providing extended precision and range for specifying the sample rate.

F0 7E id 05 05 ss ss ff <ratei> <ratef> <len> <start> <end> tt nn F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
05	Sub ID#1 = Sample Dump Extensions
05	Sub ID#2 = Extended Dump Header
ss ss	Sample number (LSB first)
ff	Sample Format (number of significant bits : 8-28)
<ratei>	Sample rate integer portion in Hertz (4 bytes, LSB first)
<ratef>	Sample rate fractional portion in Hertz (4 bytes, LSB first)
<len>	Sample length in words (0-32 GB) (5 bytes, LSB first)
<start>	Sustain Loop start point word number (5 bytes, LSB first)
<end>	Sustain Loop end point word number (5 bytes, LSB first)
tt	Loop type (see following table)
nn	Number of channels
F7	EOX

Notes :

<ratei>	Allows fixed point eight byte representation of the sample rate.
<ratef>	Specified as 28 bits for the integer part and 28 bits for the fractional part, which allow sampling rates up to ((2^28)-1) + ((2^28)-1)/2^28, or 268435455.9999999962747 Hz to be represented. This is a change from the original Sample Dump Header, which specified the sample period.
<end>	The Sustain Loop End Point is the last sample played.

Page content last updated : Mon 26 Sep 2022

<nn>	The number of audio channels (0-127) of the current sample. The total number of samples within the Sample Dump is the number of channels times the sample size. If the number of audio channels within the header is defined as 0 (nn = 0) this should be considered an error condition. Commonly expected values are nn = 1 and nn = 2. Samples for audio channels are interleaved. For stereo samples, ch1=left and ch2=right. If there are 3 audio channels (nn = 3), then the samples are interleaved as follows (where c1 = channel 1, c2 = channel 2, and c3 = channel 3): c1c2c3 c1c2c3.

Loop types

In the descriptions below, 'forward' refers to playing samples in order from lower memory addresses to higher memory addresses, whilst 'backward' playback refers to playing samples in order from higher memory addresses to lower memory addresses.

Sample pointers appear in the order : Sample start, Loop start, Loop end, Sample end.

tt	Description
00	Forward playback with a uni-directional loop. The sample plays from sample start to loop end, jumps to loop start, and continues in this fashion until voice playback stops.
01	Forward playback with a bi-directional loop. The sample plays from sample start to loop end, plays backwards to loop start, plays forwards to loop end, and continues in this fashion until voice playback stops.
02	Forward playback with a uni-directional loop and release. The sample plays from sample start to loop end, jumps to loop start, plays forwards to loop end, and continues in this fashion until key up. When the key is released, continue playing the sample to the end of the programmed loop, and then play the remaining portion of the sample after the programmed loop.
03	Forward playback with a bi-directional loop and release. The sample plays from sample start to loop end, plays backwards to loop start, plays forwards to loop end, and continues in this fashion until key up. When the key is released, continue playing the sample to the end of the programmed loop, and then play the remaining portion of the sample after the programmed loop.
40	Backward playback with a uni-directional loop. The sample plays from sample end to loop start, jumps to loop end, and continues in this fashion until voice playback stops.
41	Backward playback with a bi-directional loop. The sample plays from sample end to loop start, plays backwards to loop end, plays forwards to loop start, and continues in this fashion until voice playback stops.
42	Backward playback with a uni-directional loop and release. The sample plays backwards from sample end to loop start, jumps to loop end, plays backwards to loop start, and continues in this fashion until key up. When the key is released, continue playing the sample to the end of the programmed loop, and then play the remaining portion of the sample, after the programmed loop, backwards.
43	Backward playback with a bi-directional loop and release. The sample plays backwards from sample end to loop start, plays forwards to loop end, plays backwards to loop start, and continues in this fashion until key up. When the key is released, continue playing the sample to the end of the programmed loop, and then play the remaining portion of the sample, after the programmed loop, backwards.
7E	Backward one-shot playback, no looping.
7F	Forward one-shot playback, no looping.

As with the above Loop Point Transmission and Loop Point Request messages, the following two messages overcome the single loop shortcoming of SDS, whilst additionally allowing longer loops to be specified. Here also, loop points can be modified without having to re-send the entire sample.

Extended Loop Point Transmission

F0 7E id 05 06 ss ss nn nn tt <start> <end> F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
05	Sub ID#1 = Sample Dump Extensions
06	Sub ID#2 = Extended Loop Point Transmission

Page content last updated : Mon 26 Sep 2022

ss ss	Sample number (LSB first)
nn nn	Loop number (LSB first): 7F 7F = Delete all loops
tt	Loop type (same as in Extended Dump Header, see previous table)
<start>	Loop start address (in samples; 5 bytes, LSB first)
<end>	Loop end address (in samples; 5 bytes, LSB first)
F7	EOX

Extended Loop Point Request

F0 7E id 05 07 ss ss nn nn F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
05	Sub ID#1 = Sample Dump Extensions
07	Sub ID#2 = Extended Loop Point Request
ss ss	Sample number (LSB first)
nn nn	Loop number (LSB first): 7F 7F = Request all loops
F7	EOX

Back to index of Non Real Time Universal Sys Ex IDs

General System Information

Device Identity Request

This message is sent to request the identity of the receiving device.

F0 7E id 06 01 F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
06	Sub ID#1 = General System Information
01	Sub ID#2 = Device Identity Request
F7	EOX

Device Identity Reply

F0 7E id 06 02 mm ff ff dd dd ss ss ss ss F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
06	Sub ID#1 = General System Information
02	Sub ID#2 = Device Identity message
mm	Manufacturers System Exclusive ID code. If mm = 00, then the message is extended by 2 bytes to accommodate the additional manufacturers ID code.
ff ff	Device family code (14 bits, LSB first)
dd dd	Device family member code (14 bits, LSB first)
ss ss ss ss	Software revision level (the format is device specific)
F7	EOX

Back to index of Non Real Time Universal Sys Ex IDs

Page content last updated : Mon 26 Sep 2022

File Dump

This standard is similar to the Sample Dump Standard (SDS) but applies to file data rather than to sample data.

It comprises three similarly named messages (**Dump Request**, **Dump Header** and **Data Packet**), which as with the SDS are used in combination with the five Handshaking messages (**EOF**, **ACK**, **NAK**, **Wait** and **Cancel**) in accordance with the Dump Procedure.

File Dump Header

F0 7E id 07 01 ss <type> <length> <name> F7	
F0 7E	Universal Non Real Time Sys Ex header
id	Device ID of requester / receiver
07	Sub ID#1 = File Dump
01	Sub ID#2 = Header
ss	Device ID of sender
<type>	Type of file (4 7-bit ASCII bytes) : **Type** **DOS extension** **Description** 'MIDI' .MID MIDI File 'MIEX' .MEX MIDIEX File 'ESEQ' .ESQ ESEQ File 'TEXT' .TXT 7-bit ASCII Text File 'BIN ' .BIN Binary File (e.g. MS-DOS) 'MAC ' .MAC Macintosh File (MacBinary header)
<length>	File length (4 7-bit bytes, LSB first)
<name>	Filename (7-bit ASCII bytes; as many as needed)
F7	EOX

File Dump Data Packet

F0 7E id 07 02 pp bb <data> <checksum> F7	
F0 7E	Universal Non Real Time Sys Ex header
id	Device ID of receiver
07	Sub ID#1 = File Dump
02	Sub ID#2 = Data Packet
pp	Packet count
bb	Packet size (number of encoded data bytes - 1)
<data>	Data (encoded as below)
<checksum>	XOR of all bytes following the SOX up to the checksum byte
F7	EOX

Encoding of data :

The 8-bit file data needs to be converted to 7-bit form, with the result that every 7 bytes of file data translates to 8 bytes in the MIDI stream.

For each group of 7 bytes (of file data) the top bit from each is used to construct an eighth byte, which is sent first. So :

```
AAAAaaaa BBBBbbbb CCCCcccc DDDDdddd EEEEeeee FFFFffff GGGGgggg
```

Page content last updated : Mon 26 Sep 2022

becomes :

```
0ABCDEFG 0AAAaaaa 0BBBbbbb 0CCCcccc 0DDDdddd 0EEEeeee 0FFFffff 0GGGgggg
```

The final group may have less than 7 bytes, and is coded as follows (e.g. with 3 bytes in the final group) :

```
0ABC0000 0AAAaaaa 0BBBbbbb 0CCCcccc
```

File Dump Request

F0 7E id 07 03 ss <type> <name> F7	
F0 7E	Universal Non Real Time Sys Ex header
id	Device ID of request destination (file sender)
07	Sub ID#1 = File Dump
03	Sub ID#2 = Request
ss	Device ID of requester (file receiver)
<type>	Type of file (4 7-bit ASCII bytes) : **Type** **DOS extension** **Description** 'MIDI' .MID MIDI File 'MIEX' .MEX MIDIEX File 'ESEQ' .ESQ ESEQ File 'TEXT' .TXT 7-bit ASCII Text File 'BIN ' .BIN Binary File (e.g. MS-DOS) 'MAC ' .MAC Macintosh File (MacBinary header)
<name>	Filename (7-bit ASCII bytes; as many as needed)
F7	EOX

Back to index of Non Real Time Universal Sys Ex IDs

MIDI Tuning Standard

This standard requires that each of the 128 MIDI note numbers can be tunable to any frequency within the instrument's range. Additionally it allows provision for up to 128 tuning *programs*.

An instrument which supports MIDI Tuning may have less than the full complement of tuning programs.

The specification provides a frequency resolution somewhat finer than one-hundredth of a cent, which should be fine enough for most needs. Instruments supporting MIDI Tuning need not necessarily provide this fine a resolution – the specification merely permits the transfer of tuning data at any resolution up to this limit.

See also the Real Time MIDI Tuning messages and Registered Parameter Numbers 3 (**Select Tuning Program**) and 4 (**Select Tuning Bank**).

Bulk Tuning Dump Request

F0 7E id 08 00 tt F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)

Page content last updated : Mon 26 Sep 2022

08	Sub ID#1 = MIDI Tuning Standard
00	Sub ID#2 = Bulk Tuning Dump Request
tt	Tuning Program number (0-127)
F7	EOX

Bulk Tuning Dump Reply

This message comprises frequency data in the 3-byte format outlined below, for all 128 MIDI note numbers, in sequence from note 0 (sent first) to note 127 (sent last).

F0 7E id 08 01 tt <name> [xx yy zz] ... <checksum> F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
08	Sub ID#1 = MIDI Tuning Standard
01	Sub ID#2 = Bulk Tuning Dump Reply
tt	Tuning Program number (0-127)
<name>	Name (16 7-bit ASCII bytes)
[xx yy zz]	Frequency data for one note (repeated 128 times). See below.
<checksum>	Checksum
F7	EOX

Frequency data :

The 3-byte frequency data has the following format :

```
0xxxxxxx 0abcdefg 0hijklmn
```

Where :

xxxxxxx = semitone (MIDI note number);
abcdefghijklmn = fraction of a semitone, in 0.0061 cent units.

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 required frequency. The remaining two bytes specify the fraction of 100 cents above the semitone at which the required frequency lies.

The frequency data of **7F 7F 7F** has special significance, and indicates a *no change* situation. I.e. when an instrument receives these 3 bytes as frequency data, it should make no change to its stored frequency data for that MIDI key number.

Back to index of Non Real Time Universal Sys Ex IDs

General MIDI (GM)

Enable

F0 7E id 09 01 F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
09	Sub ID#1 = General MIDI (GM)
01	Sub ID#2 = Enable

F7	EOX

Disable

F0 7E id 09 02 F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
09	Sub ID#1 = General MIDI (GM)
02	Sub ID#2 = Disable
F7	EOX

Back to index of Non Real Time Universal Sys Ex IDs

Down-Loadable Sounds (DLS)

The **Down-Loadable Sounds** specification allows for the downloading of specific voice definitions (generally in the form of instrument wavetables), for situations which require certain voices to be available in specific locations which differ from other voice allocation modes (GM, GS, XG, etc.).

Enable

F0 7E id 0A 01 F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
0A	Sub ID#1 = Down-Loadable Sounds (DLS)
01	Sub ID#2 = Enable
F7	EOX

Disable

F0 7E id 0A 02 F7	
F0 7E	Universal Non Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
0A	Sub ID#1 = Down-Loadable Sounds (DLS)
02	Sub ID#2 = Disable
F7	EOX

Back to index of Non Real Time Universal Sys Ex IDs

Real Time Universal System Exclusive Messages

MIDI Time Code (MTC)

Full Message

System Common Quarter Frame messages handle the basic running of the system, though are inadequate when needing to fast-forward, rewind, or locating a specific point in a sequence. For these cases, the **Full Message** should be used. This encodes the full time within a single message.

After sending this message, the sender may pause to allow the receiving devices to locate to the specified point, and then resume sending **Quarter Frame** messages.

Time is considered to be *running* upon receipt of the first **Quarter Frame** message following a **Full Message**.

F0 7F id 01 01 hr mn sc fr F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
01	Sub ID#1 = MTC
01	Sub ID#2 = Full Time Code message
hr	Hours and Type : 0yyzzzzz yy = Type: 00 = 24 fps, 01 = 25 fps, 10 = 30 fps (drop frame), 11 = 30 fps (non-drop frame). zzzzz = Hours (0-23).
mn	Minutes (0-59)
sc	Seconds (0-59)
fr	Frames (0-29)
F7	EOX

User Bits

The MTC **User Bits** message is defined to be similar to the format used by SMPTE. Message bytes u1 to u8 correspond to the 32 bits provided by SMPTE/EBU binary groups 1 to 8 respectively. These nibble fields decode to four 8-bit bytes : aaaabbbb ccccdddd eeeeffff gggghhhh.

Byte u9 contains the two SMPTE/EBU Binary Group Flag bits, where 'j' corresponds to SMPTE time code bit 59 (EBU bit 43), and 'i' corresponds to SMPTE time code bit 43 (EBU bit 27).

F0 7F id 01 02 u1 u2 u3 u4 u5 u6 u7 u8 u9 F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
01	Sub ID#1 = MTC
02	Sub ID#2 = User Bits message
u1	0000aaaa
u2	0000bbbb
u3	0000cccc
u4	0000dddd
u5	0000eeee
u6	0000ffff
u7	0000gggg
u8	0000hhhh
u9	000000ji
F7	EOX

Back to index of Real Time Universal Sys Ex IDs

Notation Information

Bar Marker

If MIDI Clocks (F8) are also being sent, the bar number takes effect at the next received F8. If MTC but

no MIDI Clocks are being sent, the bar number takes effect at the next received F1 (**MTC Quarter Frame**, System Common message).

F0 7F id 03 01 aa aa F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
03	Sub ID#1 = Notation Information
01	Sub ID#2 = Bar Marker message
aa aa	Bar number (LSB first) : [00 40] Not running [01 40] -> [00 00] Count in [01 00] -> [7E 3F] Bar number in song [7F 3F] Running; Bar number unknown
F7	EOX

Time Signature (immediate)

This and the following message each use the same data format as the Standard MIDI File Time Signature Meta Event (FF 58) with extra bytes on the end to handle compound time signatures.

F0 7F id 03 02 ln nn dd cc bb [nn dd ...] F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
03	Sub ID#1 = Notation Information
02	Sub ID#2 = Time Signature message; Immediate Change
ln	Number of data bytes following
nn	Numerator of time signature
dd	Denominator of time signature (negative power of 2)
cc	Number of MIDI clocks in a metronome click
bb	Number of notated 32nd notes in a MIDI quarter note
[nn dd ...]	Additional pairs of numerators/denominators to define a compound time signature within the same bar
F7	EOX

Time Signature (delayed)

F0 7F id 03 42 ln nn dd cc bb [nn dd ...] F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
03	Sub ID#1 = Notation Information
42	Sub ID#2 = Time Signature message; Change Next Bar
ln	Number of data bytes following
nn	Numerator of time signature
dd	Denominator of time signature (negative power of 2)
cc	Number of MIDI clocks in a metronome click
bb	Number of notated 32nd notes in a MIDI quarter note
[nn dd ...]	Additional pairs of numerators/denominators to define a compound time signature within the same bar
F7	EOX

Page content last updated : Mon 26 Sep 2022

Back to Index of Real Time Universal Sys Ex IDs

Device Control

These four messages each have a channel-based equivalent.

Device control	Channel-based
Master Volume	Channel Volume (CC 7)
Master Balance	Channel Balance (CC 8)
Master Fine Tuning	Channel Fine Tuning (RPN 1)
Master Coarse Tuning	Channel Coarse Tuning (RPN 2)

Master Volume

F0 7F id 04 01 vv vv F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
04	Sub ID#1 = Device Control message
01	Sub ID#2 = Master Volume
vv vv	Volume (LSB first) : 00 00 = volume off
F7	EOX

Master Balance

F0 7F id 04 02 bb bb F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
04	Sub ID#1 = Device Control message
02	Sub ID#2 = Master Balance
bb bb	Balance (LSB first) : 00 00 = hard left, 7F 7F = hard right
F7	EOX

Master Fine Tuning

F0 7F id 04 03 tt tt F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
04	Sub ID#1 = Device Control message
03	Sub ID#2 = Master Fine Tuning
tt tt	Fine Tuning (LSB first). Displacement in cents from A440 **LSB** **MSB** **Displacement (cents)** 00 00 100 / 8192 * (-8192) 00 40 100 / 8192 * 0 7F 7F 100 / 8192 * (+8191)
F7	EOX

The total fine tuning displacement in cents from A440 for each MIDI channel is the summation of the displacement of this Master Fine Tuning and the displacement of the Channel Fine Tuning RPN.

Master Coarse Tuning

Page content last updated : Mon 26 Sep 2022

F0 7F id 04 04 00 tt F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device (default = 7F = All devices)
04	Sub ID#1 = Device Control message
04	Sub ID#2 = Master Coarse Tuning
00 tt	Coarse Tuning (LSB first, though the LSB is always zero) **LSB** **MSB** **Displacement (cents)** 00 00 100 * (-64) 00 40 100 * 0 00 7F 100 * (+63)
F7	EOX

The total coarse tuning displacement in cents from A440 for each MIDI channel is the summation of the displacement of this Master Coarse Tuning and the displacement of the Channel Coarse Tuning RPN.

Back to index of Real Time Universal Sys Ex IDs

MTC Cueing

This essentially duplicates some of the Non Real Time MIDI Cueing messages, though without the time field, as the messages described here are intended to take effect immediately.

Cueing

F0 7F id 05 tt ee ee <info> F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device
05	Sub ID#1 = MTC Cueing messages
tt	Sub ID#2 = Setup Type (see the following table)
ee ee	Event number (LSB first)
<info>	Only present for Setup Types (tt) : 07, 08, 0C and 0E (see note 1 below)
F7	EOX

tt	Description	
00	**Special** is used to control a unit as a whole (as opposed to individual sequences, sounds, etc.) **ee ee** **Description** 04 00 **System Stop** causes a unit to immediately shut down. Used to (eg) prevent a tape recorder running past the end of the reel.	
01	**Punch In**	These enable / disable record mode for the receiving device. The Event Number (ee ee) specifies the track number.
02	**Punch Out**	
05	**Event Start**	These refer to the running or playback of an event, and are used when a long sequence of events or a continuous event is to be started / stopped. Hence an event is a *macro*. The Event Number (ee ee) specifies which event is being refered to.
06	**Event Stop**	
07	**Event Start + Info**	As above, though with additional information (see note 1, below).
08	**Event Stop + Info**	
0B	**Cue Point**	This is a marker for a particular occurrence, e.g. an editing reference, or sound effects point. Each Cue Number is assigned to such an occurrence. (Use **Event Start/Stop** for continuous events.)

Page content last updated : Mon 26 Sep 2022

0C	**Cue Point + Info**	This is like **Event Start/Stop with additional Info**, except that the event represents a Cue Point rather than a Start/Stop Point. See note 1 below.
0E	**Event Name in Info**	This is used to label an Event Number. See note 1 below.

Notes :

1. Additional Information is a stream of nibblised MIDI data, LS nibble first, except when Setup Type = 0E, where it is nibblised ASCII data, again LS nibble first. An ASCII newline comprises a CR followed by a LF.

Back to index of Real Time Universal Sys Ex IDs

MIDI Machine Control (MMC) Commands

'Single-byte' MMC Commands

These messages require no additional data beyond the command code (Sub ID#2) itself.

F0 7F id 06 cc F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device
06	Sub ID#1 = MMC Command
cc	Sub ID#2 = Command (see the following table)
F7	EOX

cc	Description	
01	**Stop**	Instructs the receiving device to immediately cease playback.
02	**Play**	Instructs the receiving device to immediately begin playback.
03	**Deferred Play**	Instructs the receiving device to begin playback. If the device is busy (e.g. it has been instructed to locate the playhead to a specific position, but hasn't yet got there) then playback will begin when the device is ready.
04	**Fast Forward**	Instructs the receiving device to immediately enter fast forward mode.
05	**Rewind**	Instructs the receiving device to immediately enter rewind mode.
06	**Record Strobe**	Instructs the receiving device to immediately start recording (Punch In).
07	**Record Exit**	Instructs the receiving device to immediately stop recording (Punch Out).
08	**Record Pause**	Instructs the receiving device to immediately enter record ready mode.
09	**Pause**	Instructs the receiving device to immediately enter paused mode.
0A	**Eject**	
0B	**Chase**	
0C	**Command Error Reset**	
0D	**MMC Reset**	Resets the receiving device to its default / start-up state.

Locate / Go To

This message moves the receiving device's playhead to the specified SMPTE location.

F0 7F id 06 44 06 01 hr mn sc fr sf F7	
F0 7F	Universal Real Time Sys Ex header

Page content last updated : Mon 26 Sep 2022

id	ID of target device
06	Sub ID#1 = MMC Command
44	Sub ID#2 = Locate / Go To
06	number of data bytes that follow
01	
hr	Hours and Type : 0yyzzzzz yy = Type: 00 = 24 fps, 01 = 25 fps, 10 = 30 fps (drop frame), 11 = 30 fps (non-drop frame) zzzzz = Hours (0-23)
mn	Minutes (0-59)
sc	Seconds (0-59)
fr	SMPTE frame number (0-29)
sf	SMPTE sub-frame number (0-99)
F7	EOX

Shuttle

This message provides bi-directional shuttling of the receiving device's playhead position.

F0 7F id 06 47 03 sh sm sl F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device
06	Sub ID#1 = MMC Command
47	Sub ID#2 = Shuttle
03	number of data bytes that follow
sh sm sl	Shuttle direction and speed. Bit 6 of sh gives direction (0 = forward, 1 = backward)
F7	EOX

Back to index of Real Time Universal Sys Ex IDs

MIDI Tuning Standard

See also the Non Real Time MIDI Tuning messages.

Single Note Tuning Change

This message enables retuning of individual MIDI note numbers to new frequencies in real time as a performance control. It also allows multiple changes to be made using a single message.

Note that if the note being retuned is currently sounding (within a tone generator), the note should be immediately retuned as it continues to sound with no glitching, re-triggering or other audible artifacts.

See also Registered Parameter Numbers 3 (**Select Tuning Program**) and 4 (**Select Tuning Bank**), which allow the selection of predefined tunings.

F0 7F id 08 02 tt nn [kk xx yy zz] ... F7	
F0 7F	Universal Real Time Sys Ex header
id	ID of target device
08	Sub ID#1 = MIDI Tuning Standard
02	Sub ID#2 = Single Note Tuning Change
tt	Tuning Program number (0-127)
nn	Number of changes: 1 change = 1 set of [kk xx yy zz]

Page content last updated : Mon 26-Sep-2022

[kk	MIDI Key number
xx yy zz]	Frequency data for key 'kk' (repeated 'nn' times). See below.
F7	EOX

Frequency data :

The 3-byte frequency data has the following format :

```
0xxxxxxx 0abcdefg 0hijklmn
```

Where :

xxxxxxx = semitone (MIDI note number);
abcdefghijklmn = fraction of a semitone, in 0.0061 cent units.

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 required frequency. The remaining two bytes specify the fraction of 100 cents above the semitone at which the required frequency lies.

Back to index of Real Time Universal Sys Ex IDs

Page content last updated : Mon 26 Sep 2022