Advanced Concepts

- **Working with Transformers**
- **Working with Faders**
- **Working with Instruments**
- **Working with Audio (VST, ReWire & Soft-Synths)**
- **Using Aliases**
- **Using Macros**
- **Miscellany**

Working with Transformers

- **The Concept**
- **Message Types**
- **Transformer Modes**
- **Numerical Conditions & Operations**
- **Channel Operations**
- **Transformer Maps**
- **Transforming Notes**

The Concept

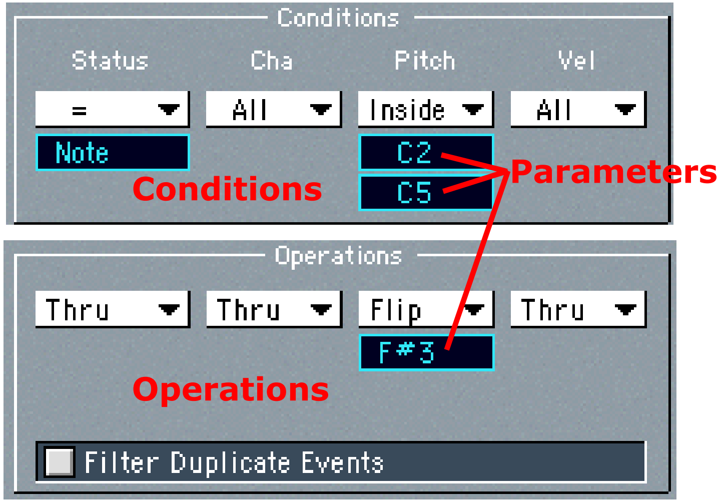
Transformers are Logic’s way of selecting and modifiying messages.

Messages are selected using conditions. The conditions specify what kind of message (‘Status’) what channel (‘Cha’) and what parameter values (‘Pitch’ or ‘-1-’ and ‘Vel’ or ‘-2-’).

Modifications are indicated by operations.You can change the kind of message, its channel and each of its parameter values.

Here notes between C2 and C5 have their pitch ‘flipped’ around F#3—the note half way between. The ascending chromatic scale from C2 to C5 will be modified to the descending chromatic scale from C5 to C2.

What happens to both the messages which are selected (notes between C2 & C5 in the example) and those which are not (everything else) depends on the ‘Mode’ menu.



Conditions

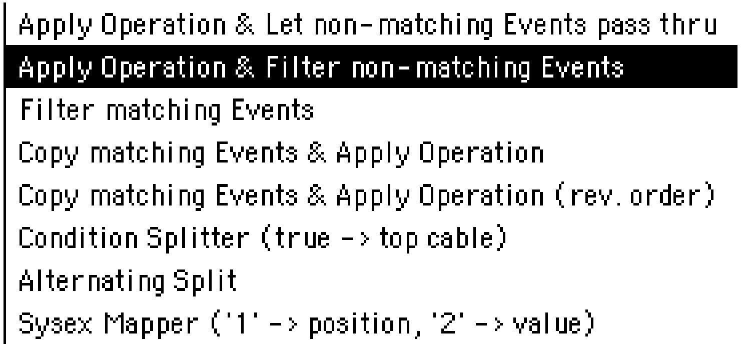
Status	Cha	Pitch	Vel
=	All	Inside	All
Note		C2 C5	

Operations

Thru	Thru	Flip F#3	Thru

□ Filter Duplicate Events

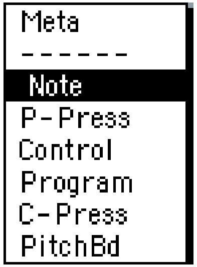
Mode Menu



Apply Operation & Let non-matching Events pass thru
Apply Operation & Filter non-matching Events
Filter matching Events
Copy matching Events & Apply Operation
Copy matching Events & Apply Operation (rev. order)
Condition Splitter (true -> top cable)
Alternating Split
Sysex Mapper ('1' -> position, '2' -> value)

Message Types

Messages come in two varieties: MIDI messages which are used by all MIDI devices and meta-messages which are used only by Logic. Transformers can select and operate on both types of messages, but some meta-messages are designed to affect Transformers. These particular meta-messages can not be selected—they perform their action instead.



For consistency, meta-messages have the same form as MIDI controller messages. They have three parameters which the Transformer labels ‘Cha’, ‘-1-’ and ‘-2-’. The ‘Cha’ parameter has no real meaning but can be used to select and route meta-messages. The ‘-1-’ parameter determines what the meta-message does (just like the MIDI controller message’s controller number) and the ‘-2-’ parameter is the message value. The meta-message numbers that affect Transformers are:

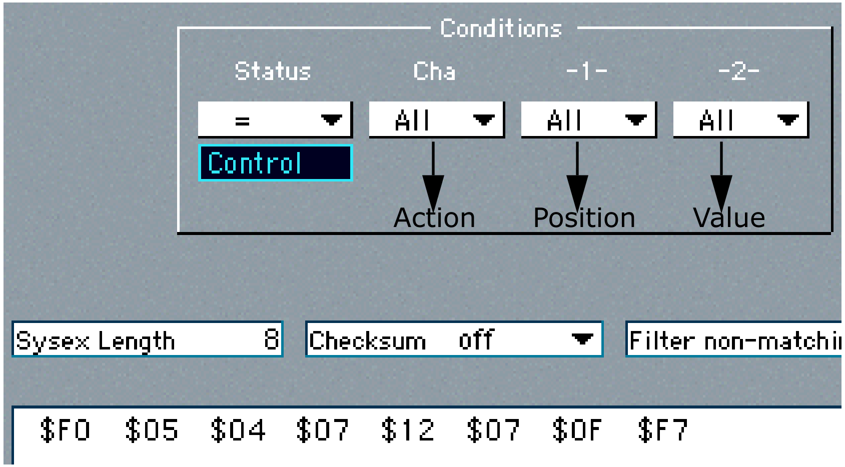
Number	What It Does
122	Sets the map value for the current map position.
123	Sets the current map position.
124	Sets the minimum (top) condition parameter for all active conditions.
125	Sets the maximum (bottom) condition parameter for all active conditions.
126	Sets the minimum (top) operation parameter for all active operations.
127	Sets the maximum (bottom) operation parameter for all active operations.
	An active condition is a ‘Cha’, ‘-1-’ or ‘-2-’ condition that is not set to “All”. An active operation is a ‘Cha’, ‘-1-’ or ‘-2-’ operation that is not set to “Thru”.

Transformer Modes

Mode	Effect on Matching Messages	Effect on Non-Matching Messages
Apply & Let non-matching...	**Changed by operation**	**Passed thru unchanged**
Apply & Filter non-matching...	**Changed by operation**	**Filtered out**
Filter matching...	**Filtered out**	**Passed thru unchanged**
Copy matching...	**Changed by operation (second)**	**Passed thru unchanged (first)**
Copy matching...(rev. order)	**Changed by operation (first)**	**Passed thru unchanged (second)**
Condition Splitter...	**Changed and sent out top outlet**	**Passed thru to bottom outlet**
Alternating Split	**No conditions or operations. Output alternates between outlets 1&2**	
Sysex Mapper...	**Sets SysEx bytes & action**	**Filtered out**

More about the SysEx Mapper Mode:

- The ‘Status’ condition selects which MIDI messages control the SysEx message.
- The ‘Cha’ controls whether the SysEx message is changed & sent (1), changed & not sent (2) or not changed & sent (3).
- The ‘-1-’ value selects which SysEx byte to change.
- The ‘-2-’ value sets the SysEx byte.
- You set the SysEx message length and checksum format.



Status: = Control
Cha: All
-1-: All
-2-: All

Sysex Length 8
Checksum off
Filter non-matchi

$F0 $05 $04 $07 $12 $07 $0F $F7

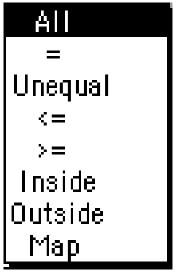
Position: #1 #2 #3 ...

Numerical Conditions and Operations

All conditions and operations except ‘Status’ are numerical.

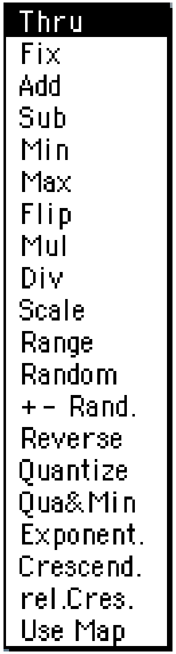
Numerical conditions compare the value of the incoming message with parameters you set in the Transformer window and select those messages which match the condition (e.g. equal to 23, between 36 and 96, etc.) Here are some handy tips:

- ‘Inside’ includes the end points whereas ‘Outside’ does not.
- ‘Map’ uses the map value of the position corresponding to the incoming value.

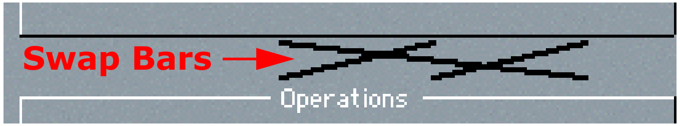


Numerical operations change the value of the incoming message according to some numerical operation and the parameters you set in the Transformer window. (e.g. add 1, multiply by 7, etc.) Here are some more handy tips:

- The ‘Flip’ operation is: 2 x parameter - incoming value.
- ‘Reverse’ is the same as ‘Flip’ with a parameter of 63.5.
- ‘Scale’ multiplies by the top parameter then adds the bottom parameter.
- ‘Quantize’ and ‘Qua&Min’ round to the nearest multiple of the parameter, but ‘Qua&Min’ ensures that the value is never smaller than the parameter.
- ‘Cresend.’ and ‘rel.Cres.’ are placebo operations.
- ‘Use Map’ takes the map value of the position corresponding to the incoming value.

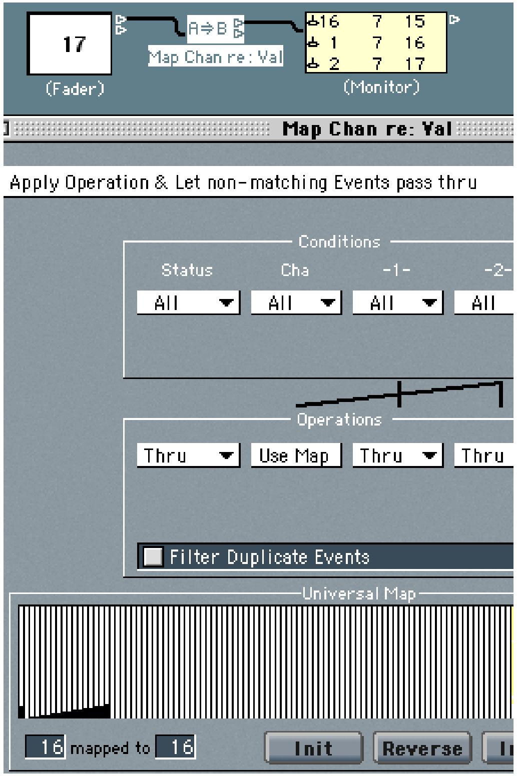


You can change which incoming value an operation applies to by using the Transformer’s ‘Swap Bars’.



Channel Operations

When performing channel operations using swapped values, Logic first ‘reduces’ the swapped value modulo 16 (i.e. takes the remainder after dividing by 16) then adds 1 before applying the operation. If you’re using a Transformer map for the operation this means that only map positions 1 thru 16 are used (position 0 is not used!). The result of the operation is the new channel number counting from 1.



16	7	15
1	7	16
2	7	17

Apply Operation & Let non-matching Events pass thru

Conditions: Status All, Cha All, -1- All, -2- All

Operations: Thru, Use Map, Thru, Thru

□ Filter Duplicate Events

16 mapped to 16

In the illustration the incoming controller values are 15, 16 & 17. These are first reduced to 15, 0 & 1 then 1 is added to get 16, 1 & 2. Map positions 16, 1 & 2 have values of 16, 1 & 2 respectively, so these become the new channel numbers. (If in a later Transformer you were to swap these back to the -2- position, you would get values 15, 0 & 1 because when converting from channel numbers, Logic subtracts 1.)

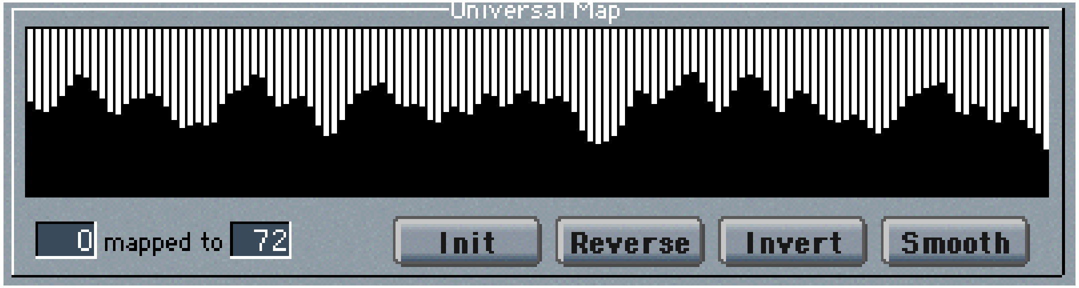
Transformer Maps

The Transformer map is just like any other operation except that you set the map values yourself—they do not need to follow any numerical formula. You can use the map for either conditions or operations.

You can set the map values by hand using the numericals. You can also set the map to match any numerical operation by selecting that it as the '-2-' operation then lock it in by changing any operation to 'Use Map' or any condition to 'Map'.

You can also use meta-messages (122 for position & 123 for value) to set the map remotely as is done in the Map Maker Tool. There is also a Transformer operation called 'MapSet' which can be used to set another Tranformer's map—it uses the '-1-' value for the map position and the '-2-' value for the map value.

You can even loop the output of a Transformer through some other processing and back into the Transformer to change its own map—see the Note Toggle tool. (Don't worry about feedback—Logic prevents it.)



0 mapped to 72

Transforming Notes

Logic applies some restrictions when applying Transformer conditions and operations to notes. These are mostly aimed at preventing hanging notes and they can be mostly gotten around. Here they are:

- **You can not apply the ‘Random’ or ‘+ - Rand.’ operation to pitch. You can construct a map using either of these operations then apply the map to pitch. You can also change the note to a control message, randomize its ‘-1-’ value then change it back to a note. Look out for hanging notes from live input if you do this.**
- **You can not set a condition to exclude notes with velocity =0 unless the ‘Status’ operation also changes the note to some other MIDI message. You can use conditions which exclude or split notes with velocity ≠0, however.**
- **You can not change a note’s velocity to 0 or change it from 0 to something else. You can change a note message to something else (e.g. a controller message) then change the ‘-2-’ value to or from 0 and finally change the message back to a note using another Transformer. (Again, be careful of hanging notes with live input.)**

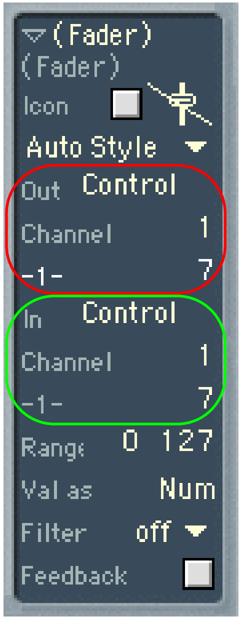
Working with Faders

- The Concept
- Fader Styles
- Fader Types
- SysEx Faders
- Vector Faders
- Meta Faders and Meta Messages
- The Fader Filter Parameter

The Concept

Faders are on-screen controls primarily for creating MIDI messages. They can create any type of MIDI or meta-message, but typically they are used for MIDI controller, program change and pitch bend messages. Less often they are used for MIDI SysEx messages and meta-messages. It rarely makes sense to use a Fader for note messages, but it is possible.

In addition to creating messages with the mouse, Faders can react to incoming messages (MIDI or meta) and they can send out different messages than they receive—so in a sense, they can act like Transformers.



▽ (Fader)
(Fader)
Icon □
Auto Style ▼
Out Control
Channel 1
-1- 7
In Control
Channel 1
-1- 7
Range 0 127
Val as Num
Filter off ▼
Feedback □

The type of message a Fader sends out is set by its 'Out' definition (circled in red in the illustration). The type of message a Fader reacts to is set by its 'In' definition (circled in green). These definitions include the message type, channel and the parameter corresponding to the Transformer's '-1-' parameter. The other message parameter (corresponding to the Transformer's '-2-' parameter) is determined by the Fader's value which can be changed on-screen with the mouse or by incoming messages matching the 'In' definition.

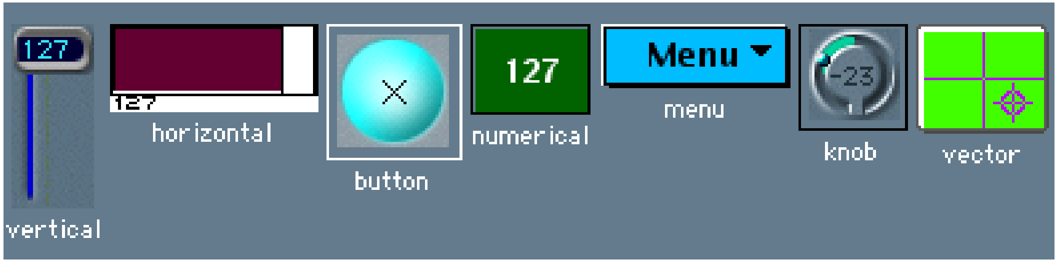
The other Fader parameters determine its look, its value range, its value display mode and its filtering properties (more later).

The 'Feedback' checkbox controls whether messages leaving the Fader can be modified and routed back to change the Fader. (Careful!!!)

Fader Styles

A Fader’s style is how it looks and how it behaves when attacked by the mouse. The seven varieties of style are shown below:

Style	How to change the Fader value
Vertical	**Click and drag vertically.** (If there is a number, you can double-click and set it manually.)
Horizontal	**Click and drag horizontally.** (If there is a number, you can double-click and set it manually.)
Button	**Click. Value alternates between left and right range values.**
Numerical	**Double-click and type in value.** (You can also click and drag vertically like a vertical Fader.)
Menu	**‘Behave as Menu’ Off: Click and drag vertically—names will scroll.** **‘Behave as Menu’ On: Click and scroll through the menu.** (Shift key keeps menu open.) **Either Mode: Double-click to open names window then click on any name to select.** (Double-click on any name to change it.)
Knob	**Click and drag any direction.** (If there is a number, you can double-click and set it manually.)
Vector	**Click and drag any direction.** (Two or four values are sent out—more below.)

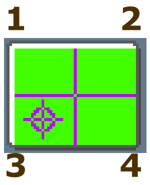


Fader Types

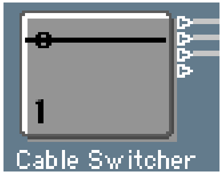
What a Fader does depends on its ‘Out’ definition. Typically it sends out messages corresponding to this definition but there are exceptions:

‘SysEx’ Faders send out one or more messages which you enter in their special ‘SysEx’ window. These include but are not restricted to MIDI SysEx messages—you can put any type of messages in there.

Vector Faders send out multiple messages. If their In and Out definitions are different the Out definition corresponds to vertical movement and the In definition corresponds to horizontal movement. If their In and Out definitions are the same then four messages are sent out on four consecutive MIDI channels starting with the channel of the ‘In’ definition. The message values for the four channels measure closeness to the corners as numbered in the illustration (assuming the ‘In’ definition uses channel 1).



Cable Switchers control the route of messages flowing into them. They sprout a new outlet each time an existing one is cabled and the Fader value controls which outlet is used. They can have any style, but the default style shown here is most indicative.

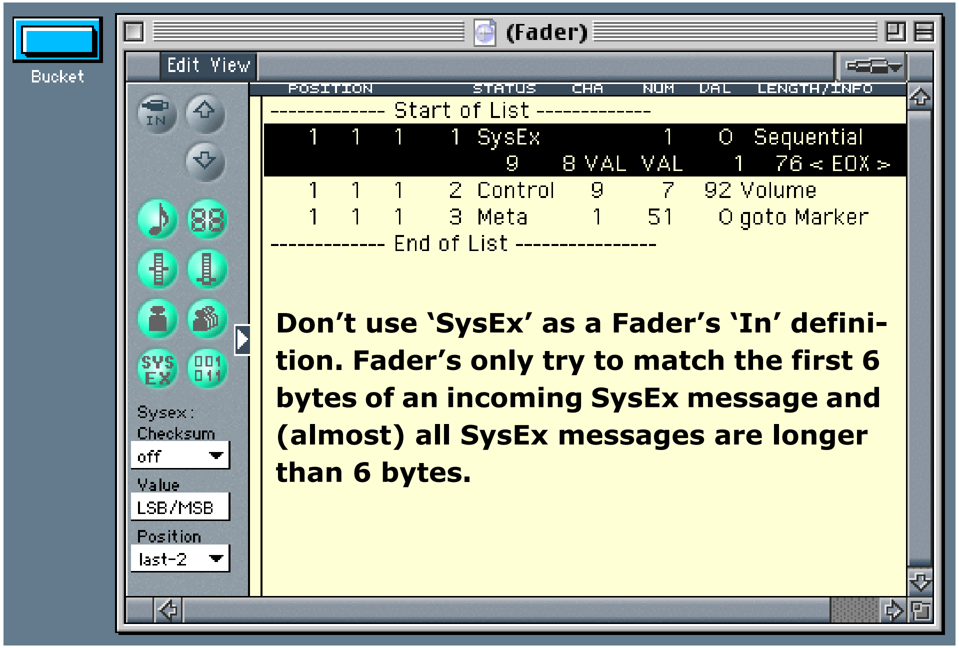


Alias Assigners are used to switch Alias Faders between different Originals. Among other things, you can use this to change a Fader’s style within a control panel. Aliases are covered in their own section below.

SysEx Faders

Although designed for MIDI System Exclusive Messages (SysEx), SysEx Faders can hold any kind of MIDI message. They make handy buckets for groups of messages like Fader group presets or device setup sequences.

When you select SysEx as an out definition a window looking like Logic's Event editor opens allowing you to create messages by Command-Clicking the green buttons. Although all the messages are sent immediately the 'Position' setting can control their order.



```
POSITION                STATUS    CHA    NUM   VAL   LENGTH/INFO
------------- Start of List -------------
   1    1    1     1  SysEx              1     0  Sequential
                         9     8 VAL  VAL      1    76 < EOX >
   1    1    1     2  Control     9       7    92 Volume
   1    1    1     3  Meta        1      51     0 goto Marker
------------- End of List ----------------
```

Sysex:
Checksum: off
Value: LSB/MSB
Position: last-2

Don't use 'SysEx' as a Fader's 'In' definition. Fader's only try to match the first 6 bytes of an incoming SysEx message and (almost) all SysEx messages are longer than 6 bytes.

Only those messages that are selected when the window is closed will have their value replaced by the Fader value. For SysEx messages you can control which bytes are replaced, what their format is and what (if any) checksum format to use.

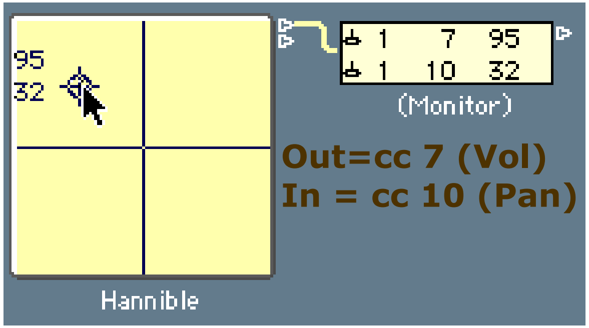
When you click the 'In' button the Fader will capture the next incoming MIDI message (no cabling required).

Vector Faders

Vector Faders allow you to control two or four parameters with a single mouse movement. When you click on the surface of the Fader, the crosshair pointer jumps to the mouse position and the numbers indicate the outgoing values—top for vertical position and bottom for horizontal position (measured from the bottom-left corner).

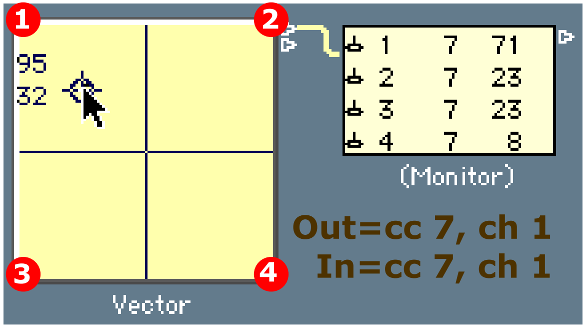
If the Fader's ‘In’ and ‘Out’ (now labeled “Vert” and “Horz”) definitions are different, two messages are output—matching the ’Out’ definition with vertical value and the ‘In’ definition with horizontal value.

If the ‘In’ and ‘Out’ definitions are the same, four message are output on consecutive MIDI channels starting with the channel of the ‘Out’ definition. The value measures the closeness to the corner (per channel as labeled). Closeness is measured by multiplying the vertical and horizontal distances from the opposite corner then dividing by 128. For corner #3: 95x32/128≈23.



```
1    7   95
1   10   32
```

Out=cc 7 (Vol)
In = cc 10 (Pan)



```
1   7   71
2   7   23
3   7   23
4   7    8
```

Out=cc 7, ch 1
In=cc 7, ch 1

Meta Faders and Meta Messages

Some of Logic’s meta-messages are designed to affect Fader parameters.

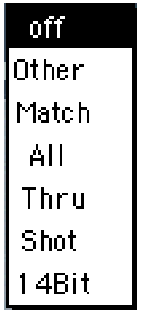
Number	What It Does
96	Sets Fader range minimum.
97	Sets Fader range maximum.
98	Sets Fader without sending value (‘Silent Fader’).
99	Sends current Fader value (‘Bang’) with or without inc., dec. and rollover: Val Action 0,1 Send current value. 120,121 Decrement before sending & roll-under from 0 to 127. 122,123 Increment before sending & roll-over from 127 to 0. 124,125 Decrement before sending & stop at 0. 126,127 Increment before sending & stop at 127. Even values cause bang message to pass thru and bang all Faders in series.

The bang message is limited to MIDI Faders—if a Fader’s ‘In’ or ‘Out’ definition is “Meta” the bang message will have no effect. The reason for this is to allow bang messages to be routed by the Cable Switcher which is actually a kind of meta-Fader.

If you want to bang a meta-Fader, use a regular Fader cabled into the meta-Fader and bang the regular Fader.

If you want to remotely step the postion of a Cable Switcher, use messages matching its ‘In’ definition with data values of 126 (decrement) or 127 (increment). Both work with roll-over.

The Fader Filter Parameter



The Fader's Filter parameter is really a multi-purpose parameter. The top five selections cause the Fader to block certain messages entering the Fader, but the last two selections change the Fader's mode of operation.

The filters ('Other', 'Match', 'All' & 'Thru') do not change how the Fader reacts on-screen—messages matching the 'In' definition will always affect the Fader. What the filter settings to control is what comes out of the Fader:

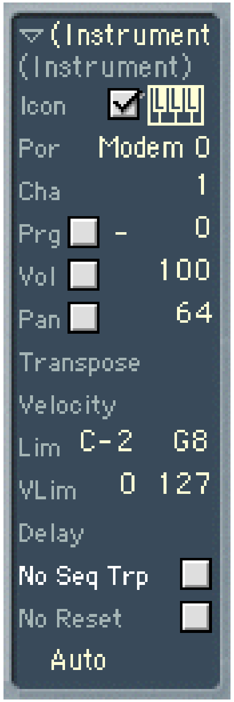
- **Off:** No Effect.
- **Other:** Non-matching messages are filtered out.
- **Match:** Matching messages are filtered out (others pass thru).
- **All:** All messages are filtered out.
- **Thru:** Messages from the Physical Input object (i.e. incoming MIDI) are filtered out. Messages created within Logic (including sequence playback) pass thru. This is how to filter out MIDI Thru when a MIDI device does not provide that option.

The Shot setting causes the Fader to send out its value only when the mouse is released—it prevents multiple values from scrolling the Fader.

The 14Bit setting causes the Fader to send out two messages for each value—one for the controller number matching the 'Out' definition and another for 32 plus that controller number. The first message carries the MSB of the data value and the second message carries the LSB. The Fader range maximum can be set as high as 16383 and if the 'Out' definition is pitch bend, an LSB/MSB pitch bend message will be sent.

Working with Instruments

Instruments are Logic’s way of managing your MIDI devices. They are one of the main ways of passing messages to your MIDI interface. In addition to having a built-in output connection, they have parameters tailored for MIDI playback devices.



▽ (Instrument
(Instrument)
Icon ☑
Por Modem 0
Cha 1
Prg □ - 0
Vol □ 100
Pan □ 64
Transpose
Velocity
Lim C-2 G8
VLim 0 127
Delay
No Seq Trp □
No Reset □
Auto

Standard Instruments

The Standard Instrument is designed for playback by a single channel MIDI device. You can control program selection (including bank), volume, pan, playback pitch & velocity limits, velocity, transpose and MIDI playback delay.

Multi-Instrument

The Multi-Instrument is designed for playback by multi-channel MIDI devices—devices that can allocate different voices and sounds to different MIDI channels. It contains 16 standard instruments, each with its own set of parameters. It also allows you to enter banks of program names and select by name.

Mapped Instrument

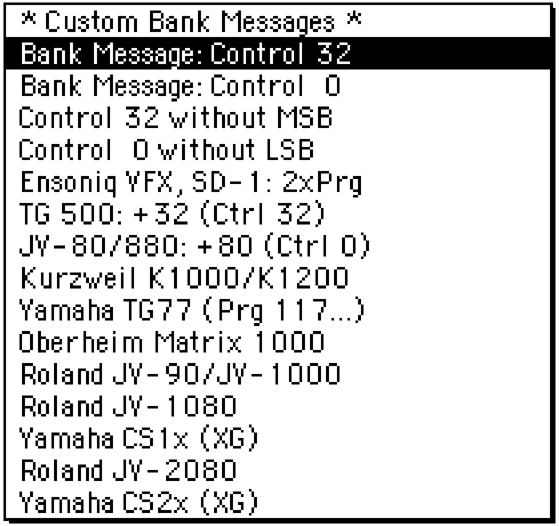
The Mapped Instrument is designed for playback by drum-mode MIDI devices. This includes playback on channel 10 of multi-channel devices ince this is often devoted to drum mode playback. There are individual parameters for each drum sound including note-mapping.

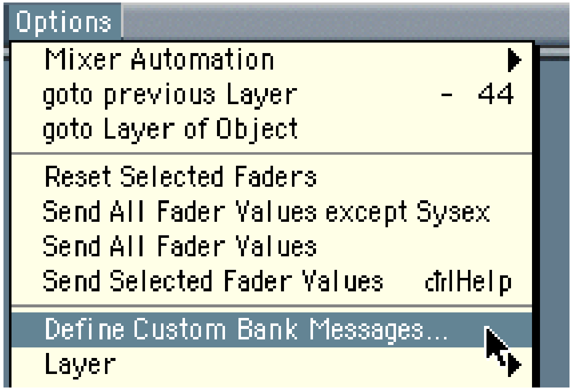
Manual: 5—24, 27, 37

Selecting Banks

The MIDI program change message is limited to 128 values. Most modern MIDI devices offer many more presets—organized in banks of 128. The standard scheme for selecting banks is to use MIDI controller #0 for the MSB and controller #32 for the LSB of the bank number. That’s the plan—reality is much different.

Manufactures use many different bank select schemes. In the Multi-Instrument, Logic has anticipated many of the common ones but for those not included and for the other Instruments, you can create your own bank select messages using the Custom Bank Select Messages window accessed from the Options menu.



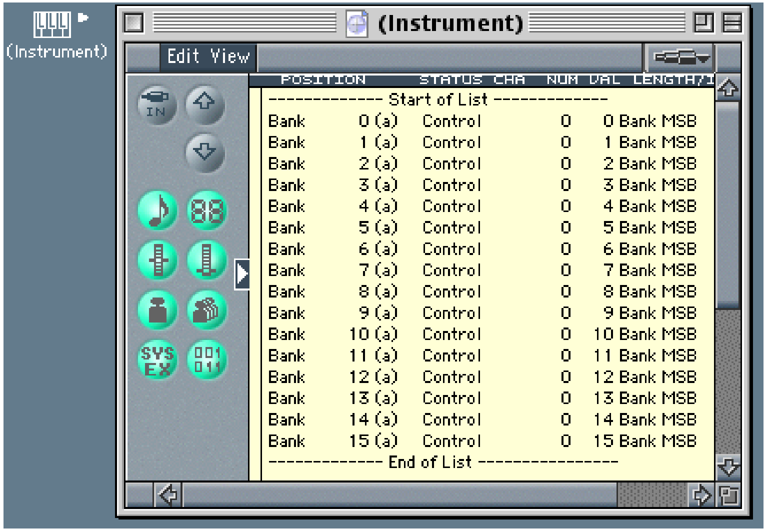


Selecting Banks (Continued)

You can create multiple select messages for any bank. The bank numbers indicate which message(s) will be used when that bank is selected. The letters in parentheses determine the message order for each bank. As with the SysEx window and the Event editor, you create new entries by Command-clicking the green buttons. Since any message type can be used, you can include device setup message when selecting banks if you wish.

Since the Multi-Instrument has only 15 name banks (0-14), you will need to use several Multi-Instruments to have access to more than 15 banks by program name. The trick is to use one Multi-Instrument for each 15 banks and create custom bank messages for each Multi-Instrument to select the correct banks.

On Arrange Tracks, assign the Multi-Instrument that includes the desired bank. For incoming MIDI (i.e. bank select automation) use the bank select numbers corresponding to the correct message in the Custom Bank Select window.



```
POSITION        STATUS  CHA   NUM  VAL  LENGTH/I
------------ Start of List ------------
Bank      0 (a)  Control       0    0 Bank MSB
Bank      1 (a)  Control       0    1 Bank MSB
Bank      2 (a)  Control       0    2 Bank MSB
Bank      3 (a)  Control       0    3 Bank MSB
Bank      4 (a)  Control       0    4 Bank MSB
Bank      5 (a)  Control       0    5 Bank MSB
Bank      6 (a)  Control       0    6 Bank MSB
Bank      7 (a)  Control       0    7 Bank MSB
Bank      8 (a)  Control       0    8 Bank MSB
Bank      9 (a)  Control       0    9 Bank MSB
Bank     10 (a)  Control       0   10 Bank MSB
Bank     11 (a)  Control       0   11 Bank MSB
Bank     12 (a)  Control       0   12 Bank MSB
Bank     13 (a)  Control       0   13 Bank MSB
Bank     14 (a)  Control       0   14 Bank MSB
Bank     15 (a)  Control       0   15 Bank MSB
------------ End of List ---------------
```

Working with Audio (VST, ReWire & Soft-Synths)

All audio routing is handled through Audio Objects in the Environment. This includes:

- **Playback of Audio Files**
  Assign audio regions to tracks assigned to standard Audio objects—i.e. Audio objects whose 'Cha' parameter is set to 'Track #'.
- **Audio Effects Buses (Send & Return)**
  Send audio to a bus using the 'Send' inserts from standard Audio objects. Route the return using an Audio object whose 'Cha' parameter is set to 'Bus #'.
- **Audio Effects Plugins (including VST)**
  Route audio through a plug-in effect using one of the audio inserts at the top of the Audio object. Track, Instrument, ReWire, Bus and Master Audio objects all support this.
- **Software Synthesizers (including VST2.0)**
  Use an Audio object whose 'Cha' parameter is set to 'Inst #' to access software synthesizer plug-ins. Route MIDI playback to the soft synth by playing it back on tracks assigned to the Audio object. Route live MIDI to the soft synth by selecting a track assigned to its Audio object. Route MIDI from the Environment to the soft synth by cabling it into the Physical Input object (through the NOF Restorer tool if necessary). Record the audio output of the soft synth using the Master Audio object's 'Bounce' function.
- **Audio Input from ReWire Applications**
  Use an Audio object whose 'Cha' parameter is set to a ReWire bus to route audio output from a running ReWire application back into Logic. (Launch Logic before the ReWire application. Logic's transport must be running to hear incoming ReWire audio.) Use a separate means (e.g. an Internal object, OMS, etc.) to route MIDI data to the other application. Use the Master Audio object's 'Bounce' function to record the output of the ReWire application in Logic.

Soft synths come in many flavors, but they all share two requirements: you must get MIDI to them and you must do something with their audio output.

MIDI In

If the soft synth is a stand-alone application, you get MIDI to it either by using an Internal object (if there is one for it) or using some MIDI management application like OMS or Hubi's Loopback.

If the soft synth is a plug-in (e.g. internal to Logic like the eS1 or a VST2.0 instrument), you can get MIDI to it only through Arrange tracks—i.e. you can't cable MIDI directly into it from other Environment objects. You can however, cable other Environment objects into the Physical Input object. This will route the output of those objects to the selected track and thus to the soft synth if its track is selected. But, this can lead to hanging notes—see the NOF Restorer tool.

Audio Out

If the soft synth is either a plug-in or a ReWire device, its audio output will be played in Logic by an Audio object (when Logic's transport is running). The only way to record this playback is using the Master Audio object's 'Bounce' function.

If the soft synth is a stand-alone device, you will need to use whatever recording provisions it offers to capture its output (i.e. recording direct to disk or recording to RAM and saving to disk.)

VST Plug-ins

VST is a protocol developed by Steinberg for implementing audio effects and soft synthesizer plug-ins. VST audio processing effects appear on the ‘Inserts’ menu of Audio objects and their counterparts in the Track Mixer. VST software synthisizers appear on the top menu (above the Inserts menu) of Audio objects designated as Instruments. (This amounts to selecting one of the ‘Instrument #’ choices from their ‘Cha’ parameter menu.)

VST plug-ins must reside inside a folder named ‘VstPlugIns’ in the same directory as the Logic application. (Logic’s built in plug-ins do not reside in any folder, but presets for them are stored in the folder named ‘Plug-In Settings’.)

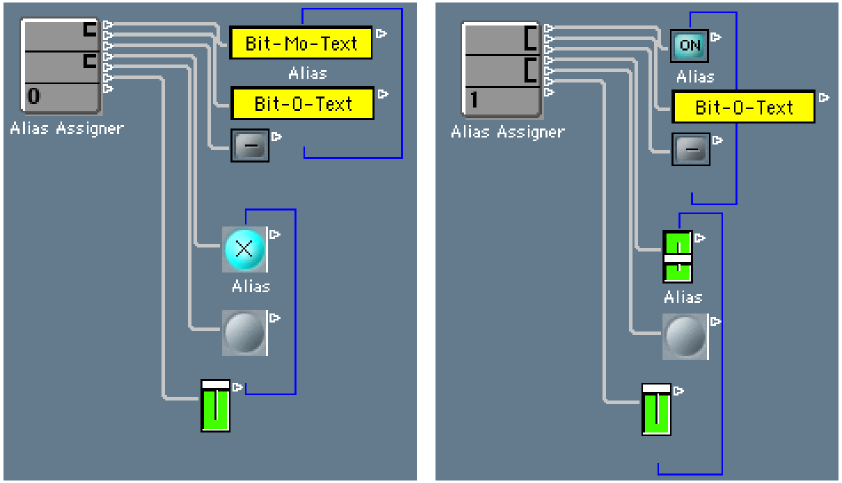
MIDI controller messages can be used to change plug-in parameters (by cabling or track playback). MIDI controllers starting with number 64 are used for parameters in the order they appear in the ‘Controls’ view of the plug-in panel. If there is more than one plug-in, 16 controllers are devoted to each (i.e. 64-79 to the top plug-in, 80-95 to the second, etc.)

For Audio objects used as soft synths, controller numbers 64-79 are always reserved for the soft synth (the top plug in) regardless of whether one is actually assigned. To make matters even more tricky, controller 64 is reserved for ‘Sustain’ and therefore is not passed through to the first soft synth control. This makes the first soft synth control unreachable via MIDI. (Pray it is assigned to an unimportant parameter!)

Using Aliases

Aliases are clones of regular Environment objects with some of their own personality traits. Aliases save some memory—especially aliases of text-style Faders. They also provide (together with the 'Alias Switcher' Fader) a clever way to change the style of control panels. But, aliases can not be used in Macros so there is a trade-off.

The illustration shows the Alias Assigner in action. The top object in each of the groups of 3 objects is the Alias and the two objects below it are Faders. The Alias Switcher switches the 'original' for each of the Aliases. With the Alias Switcher in the 0 position (left) the Alias corresponds to the middle Fader in each group. With the Alias Switcher in the 1 position, the Alias corresponds to the bottom Fader. But, notice that the Aliases can have different values. (They can also have different channels.)



For this multiple-switching technique to work the Alias Assigner's range must be '0 n' where n is the number of originals in each group.

Manual: 5—63, 82

Using Macros

Macros are one of the Environments most powerful features. They allow you to combine collections of objects and cables into a single Environment object which you can thereafter treat just like any other object. Here are some of their advantages:

- **They are easy to transfer between Songs.**
- **They can be used in other processes thereby reducing complexity.**
- **They are easily copied—to save the control settings in a Macro, simply copy it.**
- **You can double-click some objects (Transformers, text Faders, Chord Memorizers) inside Macros to change their parameters.**
- **They can be unpacked for editing or protected so that they can not be 'unpacked' to reveal the original construction.**

And some limitations:

- **They are limited in size (typically to 100-200 objects).**
- **They have a single input and output (i.e. all input goes to the same 'Macro-In' object inside the Macro and all output comes from the same 'Macro-Out' object.)**
- **Instruments in Macros lose a lot of their functionality—Multi-Instruments lose access to their sub-channels & Mapped Instruments lose their map window.**
- **You can't select and change multiple Faders in a Macro.**

Creating a Macro

You create a Macro from a collection of Environment objects and their cables by selecting all the objects and choosing 'Macro' from the New menu. Here are some things to remember:

- **Any cables to or from unselected objects will be lost. In most cases, Logic will announce this and create a Macro out of a copy of the selected objects. (Otherwise the Macro replaces the selected objects.**
- **All input to the Macro will go to the top-left object unless there is an object named 'Macro-In'.**
- **All output from the Macro will come from the bottom-right object unless there is an object named 'Macro-Out'.**
- **Any cables from the output object will be lost even if they lead to other objects in the Macro.**

Sooner or later you will try to create a Macro and get the dread "Too many objects" error. Here are some things to try in order of increasing drasticity:

- **Design out any Instrument type objects.**
- **Delete all ornaments & text faders used only as labels.**
- **Delete neutral objects (Monitors, neutral Transformers, etc.) by cabling around them.**
- **Make sure all text Faders have the smallest possible range.**
- **Replace text Faders by regular Faders.**

Miscellany

- **Working with Notes**
- **Splitting Note-ons & Note-Offs**
- **Creating Cable Junctions**
- **MIDI message format (MIDI, SysEx, NRPN, etc.)**

Working with Notes

MIDI note messages come in pairs: Turn the note on (‘Note-On’ or ‘NON’) and turn the note off (‘Note-Off’ or ‘NOF’). If you change the pitch or channel of a Note-On message without making the same change to the matching Note-Off message, the note will be left ‘hanging’.

Note-On messages with velocity 0 can be substituted for Note-Off messages—Logic does this. If you change the velocity of Note-On message from 0 to something else, the note will also be left hanging. If you change the velocity of a Note-On message to 0 from something else, the note will be ‘dropped’ (i.e. will not be played).

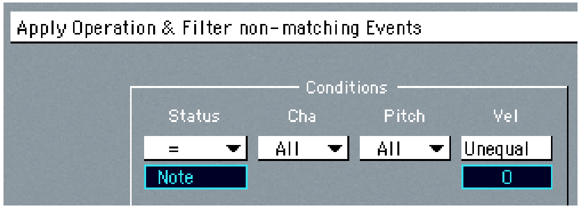
Notes are recorded in Logic as single events—as note and length. Logic makes it virtually impossible to separate these so you don’t need to worry about leaving hanging notes in playback processes. Although Logic also tries to prevent hanging notes for live MIDI input, it is still possible—for input processing you need to be careful not to leave hanging notes.

If you need to separate sequenced note-ons from note-offs, you can do so by playing the sequence back through the Physical Input object (i.e. assign it to a Track and select another Track as the output destination). All notes will be left hanging if you do this, but this might be useful if you’re using some algorithmic process to generate separate note-offs. The same applies to playback from the Arpeggiator and Delay Line objects—passing them through the Physical Input strips the note-offs.

Splitting Note-on and Note-off Messages

Some Transformer conditions and operations are automatically overridden to prevent hanging notes. The illustration below shows what you can and can not do.

Will not filter out NOFs

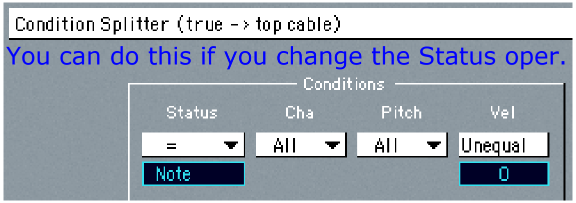


Apply Operation & Filter non-matching Events

Conditions

Status	Cha	Pitch	Vel
= Note	All	All	Unequal 0

Will not split NONs from NOFs



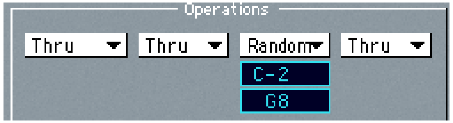
Condition Splitter (true -> top cable)

You can do this if you change the Status oper.

Conditions

Status	Cha	Pitch	Vel
= Note	All	All	Unequal 0

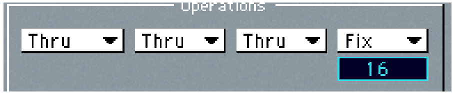
No effect on notes



Operations

Thru	Thru	Random C-2 G8	Thru

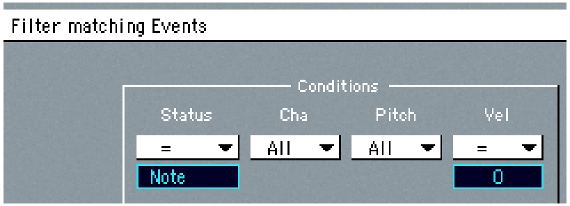
No effect on NOFs



Operations

Thru	Thru	Thru	Fix 16

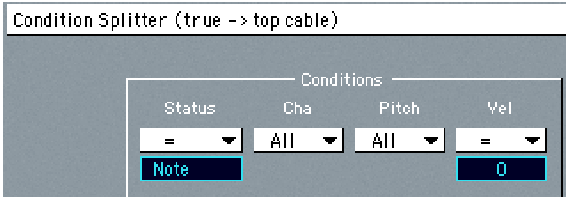
Will split NONs & NOFs leaving hanging notes



Filter matching Events

Conditions

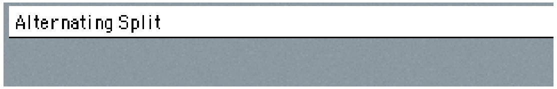
Status	Cha	Pitch	Vel
= Note	All	All	= 0



Condition Splitter (true -> top cable)

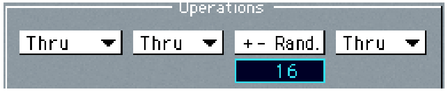
Conditions

Status	Cha	Pitch	Vel
= Note	All	All	= 0



Alternating Split

Randomized pitch offset leaves hanging notes



Operations

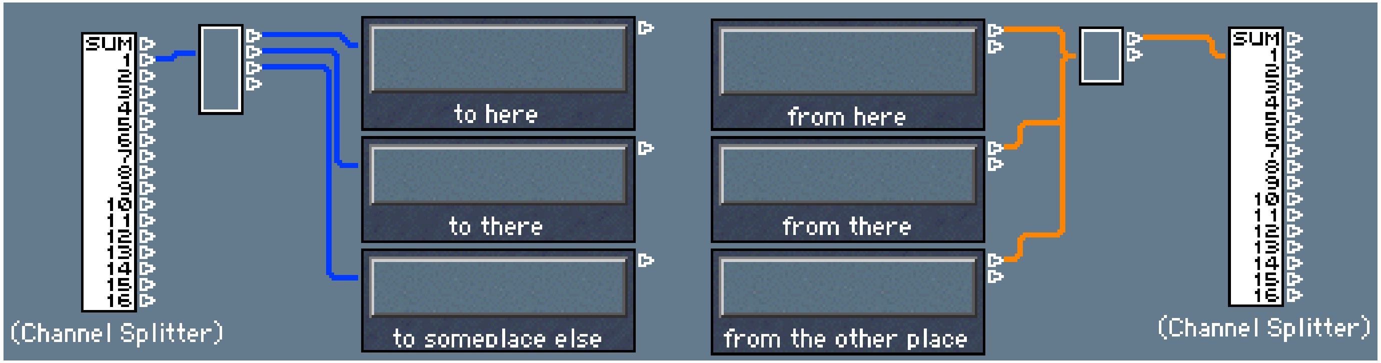
Thru	Thru	+- Rand. 16	Thru

You can usually work around these restrictions (see links buttons below).

Creating Cable Junctions

There are many occasions when you want to split one cable into two or more without doing any processing. For objects whose outlets carry different information (e.g. Channel Splitter, some Transformer Modes, Cable Switcher, etc.) you will need to do this to pass the same information to several destinations. In situations where you may want to replace one object by another it is convenient to both combine all inputs and use a single output. In other cases, it simply makes the cabling clearer.

You can use any neutral object for a cable junction, but four that work particularly well are Transformers with no condition or operation settings, Monitors, Ornaments and Faders. The Monitor has the advantage of showing what's passing through. The Ornament has the advantage of being resizable. The Transformer offers the most flexible options for later introducing processing at the junction. Faders make another reasonable choice when you want graphically display what's passing through. Also their filter conditions can later turn out to be useful.



MIDI Message Formats (MIDI, SyeEx, NRPN, etc.)

In case you want or need to know the data structure of MIDI messages, here’s a very brief summary.

By the Numbers

MIDI information is transmitted serially in ‘bits’. A bit is either high or low representing a ‘1’ or a ‘0’. Eventually it all comes down to numbers and convenient ways of naming them.

Whatever the ‘system’, names for numbers start with a base alphabet of symbols, stringing these together to name numbers. The number of symbols in the alphabet and the number of ‘digits’ in the string determine how many numbers can be named. With one digit, you can name as many numbers as you have symbols. Each time you add another digit you multiply the number of names by the number of symbols. In our common decimal (10 symbol) system there are ten 1-digit numbers (0, 1, 2, ... 9), 100 (10x10) 2 digit numbers, 1000 three digit numbers, etc.

The most natural alphabet for MIDI (and all other digital considerations) contains two symbols: 0 and 1. So there are two 1-digit numbers, four 2-digit numbers, eight 3-digit numbers, etc.

MIDI allows 8 digits and there are 256 8-digit numbers. MIDI uses half of these—those with 1 as the leftmost digit—to send commands (called ‘status bytes’) and the

remaining 128 8-digit numbers to send data. This is why the typical MIDI data range is 0-127.

Before leaving this subject we should take a look at another common system—the hexadecimal (hex) system of 16 symbols (0-9, A,B,C,D,E and F). In this system it takes two digits to name 256 numbers and the MIDI messages are divided between those whose leftmost digit is 8-F (the status bytes) and those whose digits are 0-7 (the data bytes). This is important for two reasons: it makes it easy to see how MIDI handles channels and you will often run into it when dealing with SysEx messages (e.g. in the technical manuals for your various MIDI devices).

The MIDI status bytes starting with 8 thru E are used for MIDI 'channel' messages (note-on, note-off, poly-pressure, controller, program change, aftertouch and pitch bend in that order). These are the typical MIDI messages you deal with in sequences. Since hex messages are two digits, there are 16 of them starting with each of these symbols. This is how MIDI represents the 16 MIDI channels (i.e. it is why there are only 16 channels). The left digit tells what kind of a message it is and the right digit tells what channel.

This leaves the MIDI status bytes starting with F. These are for various kinds of 'System' messages. System Exclusive (SysEx) messages are all we're going to deal with here.

SysEx messages are used to communicate to specific device models—they are 'exclusive' to these models. Typically they are used to change parameters that are not

accessible through standard MIDI channel messages (e.g. controller messages). They are also used for transferring individual presets and preset banks and for selecting among different internal configurations. In short, they are used for almost anything.

Unlike other MIDI messages which must have a specific form and length, SysEx messages don’t have to follow any rules—manufacturers are free to format and use them in any way they like with these exceptions: they must always start with $F0, the next 3 or 4 bytes must identify the manufacturer and model and they must always end with $F7. (Hex numbers are usually preceded by ‘$’ to avoid confusion.)

MIDI Channel Messages

Each of the MIDI channel messages consists of a status byte (128-255) and one or two data bytes (0-127). (See the Reference section for details.) Program change and after touch messages have one data byte and are therefore the most economical. The rest of the channel messages have two data bytes—pitch & velocity for notes, controller number & value for control messages, pitch & pressure for poly pressure, fine & coarse amount for pitch bend.

For greater efficiency (timing & bandwidth is always at a premium with MIDI) a shortcut called ‘running status’ allows the status byte to be dropped if it hasn’t changed. For example when playing a chord on the same MIDI channel, you need one status byte to say ‘these are notes’ followed by two data bytes for each note & velocity in the chord.

Although there are separate status bytes for note-on and note-off messages, another shortcut allows note-on messages with velocity 0 to be interpreted as note-off messages. Since the most common MIDI message is notes and typically there are streams of note-on/note-off pairs, this together with running status saves a lot of MIDI bandwidth.

Extending the MIDI Data Range

Individual data bytes must be between 0 and 127, but there’s no reason why only one data byte can be used to set a data value.

Pitch bend is an example of a MIDI channel message that uses two data bytes providing 16384 values (128x128). Since pitch bend needs to go both ways, the first half of the values (0-8191) are interpreted as negative pitch bend and (8193 to 16383) are interpreted as postive pitch bend. (Value 8192 is interpreted as no pitch bend.)

In general when higher resolution is needed for MIDI controller values, the controller number for the least significant part is 32 greater than the controller number for the most significant part. For example MIDI controller #7 is used for volume, but if a MIDI device needs higher resolution for volume, cc#7 would be used for the high part and cc#39 would be used for the low part.

Program bank select messages are another example where 128 values is often not enough. In this case, MIDI controller #32 and #0 are used together. Typically cc#32 is the low part of the number and cc#0 is used for the high part, but here the rules mean almost nothing—MIDI manufactures dream up the most arcane possible bank select schemes.

In the case of SysEx, such nice uniform conventions are tossed completely out the window—the stream of data values between the device identifier and the end of the SysEx message might be considered one long string of 0’s and 1’s to be broken up in any way the manufacturer sees fit.

Extending the Number of MIDI Controllers

As MIDI devices get more complex, 128 different controllers (indicated by the first of the two data bytes in a controller message) are often not enough. A similar scheme is used to extend the number of controllers using ‘Registered’ (RPN) and ‘Non-Registered’ (NRPN) Parameter Numbers. In this case six MIDI controller messages are used: 2 for the parameter number, 2 for the data value and 2 more to say you’re finished. (The last two are often not necessary.) Details can be found in the Reference section.

Summary

You will most often deal with MIDI channel messages in which the status (first) byte tells the message type & channel and the next byte or two communicates the value. In cases where the data value range of 0-127 is not enough, two data bytes are used extending the range to 0-16383. This might be within the same MIDI message as with pitch bend or using two MIDI messages as with MIDI bank select messages. In the case or RPN and NRPN messages, separate pairs of messages are used to select the MIDI device parameter and send the parameter value. With SysEx messages anything goes.