

# DSPaudio iCE

## User Manual

Document Revision 1.2

December 14, 2007



**Design, Development, Documentation, QA, Graphics, Marketing, Sales, Chief cooks, Bottle-washing:  
Anthony, Peter, Erik, Friends  
©2004-2008 DSPaudio, Inc.**

**Contact:  
email: [signal@dspaudio.com](mailto:signal@dspaudio.com)  
web: [www.dspaudio.com](http://www.dspaudio.com)**

## What are the iCE Tools?

iCE in combination with Max/MSP creates an incredibly flexible, customizable and powerful sequencing, processing, looping, sampling, multi-tracking and effects software. iCE externals greatly enhance Max/MSP opening up new forms of media-sequencing. Many new angles of timeline composition are now possible:

- Sequencing: step, timeline, tracking, looping, iterative, generative, recursive, (float/int/list/messages/hertz/etc.)
- Interfacing: fully user defined keyboard, mouse and joystick capability
- UI Building: open framework for user defined control structures

## Do I need to read the Documentation?

As long as you are familiar with Max/MSP, you will find most of the iCE Tools easy to use, providing obvious functions that you may have wished for in Max. For this reason, most objects are completely covered by the information in their \*.help files. However, the flagship object, ice.lattice, provides so many features and functions that a complete documentation is beyond the scope of the ice.lattice.help. In the following pages you will find all the information you need to build your own tracking and sequencing tools with iCE.



## What's in the iCE Package?

ice.lattice	Sequencing, tracking, data visualization and storage
ice.pump	Flexible message router
ice.key	Capture all keystrokes. Even when Max is in the background
ice.mux	An active message packer. Like pack but preserves i/o map order with nulls
ice.demux	An active message unpacker. Preserves i/o map order with hidden nulls

## Installing iCE:

### Installation is simple:

- copy the iCE\_help folder into MaxMSP/max-help/
- copy the ice.lattice-insp.pat file from the iCE\_inspector folder into MaxMSP/patches/inspectors/

The last step is to match the iCE objects to the version of Max/MSP you are using.

### For Max/MSP 4.6 and greater use the iCE\_UB folder:

- copy ice.lattice.mxo from the iCE\_UB/iCE\_startup/ folder into MaxMSP/Cycling '74/max-startup/
- copy the iCE\_objects folder into MaxMSP/Cycling '74/externals/

### For earlier versions of Max/MSP (4.2-4.5) use the iCE\_PPC folder:

- copy ice.lattice from the iCE\_PPC/iCE\_startup/ folder into MaxMSP/Cycling '74 folder alias/max-startup/
- copy the iCE\_objects folder into MaxMSP/Cycling '74 folder alias/externals/

Now start up Max/MSP and notice the new lattice icon on the object palette.

Have fun



## iCE.key

### Description:

iCE.key is a general purpose keyboard input object which supports:

- Input when Max is not the foreground application
- Input when not in the front window
- Input independent of other UI objects having “focus”
- Receive notification of any combination of key down, key up, and auto-key events
- Optionally receive notification of “dead” keystrokes (option-E, etc.) that normally do not generate key events until a second key is hit
- Optionally receive notification of modifier keys as independent keystrokes
- Optional special keystroke to report even when ice.key is deactivated.

### Typical Uses:

Outlets 1, 2 & 3 of iCE.key are packed and sent to pump for controlling an array of lattices or other UI objects. Key can also be used for fun applications like capturing passwords.



### Installation Redux:

Place the iCE.key object in your externals search path. It is recommended you make a “ice” directory in your externals folder to contain non-UI DSPaudio Max/MSP externals (typically: ice.key, ice.pump, ice.mux, ice.demux).

### Instantiation:

iCE.key <int> - iCE.key supports an optional ASCII keycode argument for “privacy.” The privacy key allows iCE.key to be deactivated and re-activated directly from the keyboard. If you want to specify the privacy key as a keyboard “scan” code instead of the ASCII character (for instance, to use one of the F-Keys), add 65536 to the scan code and use that value.

### Inlet definitions:

**INLET 1:** ASCII key codes are accepted. Use this for keystroke playback, UI control, joystick input & whatever else you can think of.

### Outlet definitions:

- OUTLET 1:** ASCII code of keystroke (when modifier key detection is on, -1 indicates modifier key)
- OUTLET 2:** Key “scan” code of keystroke
- OUTLET 3:** State of modifier keys
- OUTLET 4:** Type of keystroke (1 = key down; 0 = key up; -1 = auto-key)
- OUTLET 5:** Bang when privacy key detected

## Messages:

### **mode <int>**

The mode message specifies which keystrokes the iCE.key object will report. The integer argument can be the sum of any of the following values:

- 1**        report key down
- 2**        report key up
- 4**        report autokey
- 8**        report modifier keys when auto/up/down occur
- 16**      “invigorate” dead keys (immediately report things like option-e)

If a “mode 0” message is sent, the iCE.key object will be effectively disabled. However, a bang will still be sent through the rightmost outlet whenever the privacy key is detected.

## Keystrokes:

All.



## iCE.pump:

### Description:

Pump is a general purpose message router. It can route incoming messages to a maximum of 2520 outlets (*lately it appears limited to 255*). Of the 2 INLETS pump provides, INLET\_1 messages are evaluated for navigational key-codes. Messages not containing reserved navigation key-codes are passed to the active outlet(s) without intervention. INLET\_2 messages are unevaluated and passed to the active outlet(s).

### Typical Uses:

As a navigation interface between iCE.key and an array of iCE.lattices or max UI objects.

### Installation Redux:

Place the iCE.pump object in your externals search path. It is recommended you make an "iCE" directory in your externals folder to contain non-UI DSPaudio Max/MSP externals (typically: ice.key, ice.pump, ice.mux, ice.demux).



### Instantiation:

iCE.pump <int> - <int> is the number of outlets you would like (range is 2 – 2520). If not specified, pump will default to two outlets. An optional second argument given as a (named symbol) sends iCE.pump statistics and runtime information to a receive object of the same name. Use this feature for directing statistics and other iCE.pump information to an LCD to make your patchers more glamorous.

### Inlet definitions:

Inlet 1: Messages - evaluated for specific ASCII keyboard codes (see keystroke section towards bottom of iCE.pump section)

Inlet 2: Messages - not evaluated, passed directly to active outlet(s).

### Outlet definitions:

Incoming messages are passed through the currently active outlet(s). This includes all messages in the right inlet, and any messages in the left inlet that do not have special meanings for pump.

## Pump Keystrokes:

CursRight	self next	next outlet
CursLeft	self prev	prev outlet
CtrlCursUP	pattern kLongMax	indicates next pattern
CtrlCursDown	pattern kLongMin	indicates previous pattern
CtrlGT	octave kLongMax	indicates up one octave
CtrlLT	octave kLongMin	indicates down one octave
CursUP	goto kLongMax	indicates next row
CursDown	goto kLongMin	indicates prev row
CtrlCursRight	cursor kLongMax	indicates nudge value up
CtrlCursLeft	cursor kLongMin	indicates nudge value down
ShiftCursUp	selstart kLongMin	indicates expand selection up
ShiftCursDown	selend kLongMax	indicates expand selection down
CtrlSpace	store null	flag sends zero out middle outlet
CtrlL	lin	linear interpolation
CtrlP	pow 2.0	parabolic interpolation
CtrlE	exp 2.71828	exponential interpolation
CtrlF	type 0.0	typecast to float
CtrlI	type 0	typecast to int, truncate
CtrlR	type 1	typecast to int, round
Bspace	del	clear selection
Clear	del	synonym – clear selection
FwdDel	del	synonym – clear selection
CtrlX	cut	cut selection
CtrlC	copy	copy selection
CtrlV	paste	paste selection
Enter	bang	bang row
Return	next	same as bang
ShiftReturn	prev	indicates bang and goto prev row

**Note:** Any other 3-element list (keystroke) in iCE.pump's left inlet is passed on as a char message (with three integer arguments) so iCE.lattice or other connected object can do it's own character mapping.



## ICE.LATTICE :

### Summary & Purpose:

Lattice is an **event sequencer** that supports storage, navigation and recall (playback) of FLOATs, INTs, TEXT Messages, LISTS and META-commands organized in a visual heirarchy. It is best to think of each lattice as a multidimensional list or type limited database which puts the user in direct interaction with the contents.

The design is biased for timed sequential storage and recall although asynchronous navigation and I/O are provided for. This concept and the resultant workflow make programming storage structures and event sequencers less work and more fun.

Databases traditionally require query structures and user interfaces be created to store and recall data, but in the case of Lattice these elements are inherent to it's visual display and built in navigation. In most cases complex query & display structures are unnecessary.

Lattice is at home acting as a timeline for any application requiring a flexible event sequencer. Due to it's storage and data representation methods it is excellent for a wide variety of tasks that previously required extensive programming to accomplish.

### Common uses of Lattice by the creators:

Triggering, controlling, performing, scaling, tuning, storing & recalling Max/MSP events in both timed and arbitrary structures.

Sequencing music

Scripting Max/MSP

Interacting with external command languages such as EXPECT / TCL to accomplish automated login's and data maintainance/movement tasks on OS X or Unix variants (OS X/Linux/BSD/Solaris).

Creating various flavors of Trackers (powerful vertical music sequencers)

Browsing, Loading & Managing files similar to the EMACS file browser (much faster than OS GUIs)

Triggering and then displaying network security scanner results to create a common workflow for penetration testing.

Re-inventing the common to-do list and other silly utility functions in order to never leave Max/MSP.

Enlightenment

Lattice makes it much easier to achieve sequences incorporating: algorithmic, generative, fractal, microtuning, variable clock relationships, embedded structures, iterative, evolutionary and traditional linear sequencing. With every lattice function mapped to keyboard shortcuts (based on the 20 year tradition of trackers, continued by DSPaudio) note entry and control is optimized for video game like speeds.





## Specifications:

Lattice organizes data into sets of rows called patterns. Each row can contain anything a Max message box can contain: integer values (ints), floating-point numbers (floats), symbols, lists, and more complex messages consisting of a symbol followed by up to 255 parameters. Lattice allows you to concatenate multiple messages into a row; separate the messages by commas (just like the message box). You can specify that row messages are sent to a receive object; prefix the message with a semicolon (just like the message box). In addition, special messages encoding timing information (delays and repeats) can be stored in lattice rows. There is always an active pattern, this is the one that you see on screen, but a lattice can hold up to 8192 patterns. Each pattern can contain up to 2,148,728,832 rows. However, you should be aware that Max/MSP can currently handle a maximum of 2GB of memory.

Lattice is typically used to manage MIDI and other sequencer data, so you will find many messages geared towards easy entry and processing of MIDI and frequency data. And, although lattice resembles coll as repository for arbitrary data, lattice uses compact and efficient data storage techniques optimized for use as a sequencing machine.

## Installation:

Lattice is a DUI object. As such you must drunkenly place it in the max-startup folder. It will appear on the Max/MSP GUI toolbar upon restart.

## Instantiation:

Drag lattice off the UI toolbar and place it wherever desired. Lattices can be cut/copy/pasted without losing their respective data.

## Inlet definitions:

<b>INLET 1:</b>	Keyboard control messages from pump are evaluated. All other data is sent to the active row for storage
<b>INLET 2:</b>	Raw input is passed to row for storage without evaluation
<b>INLET 3:</b>	Integer tells lattice which pattern to make active

## Outlet definitions:

<b>OUTLET 1:</b>	INT / FLOAT / MESSAGE / LIST (typical use is MIDI Pitch, see *microtracker)
<b>OUTLET 2:</b>	ON/OFF (0 or 1). When an event (note) occurs outlet status goes high. When an event_off (note off) occurs status drops low (0).
<b>OUTLET 3:</b>	Timing index: val1,val2,val3,val4 (Row #, Pattern Length, Major interval, Minor Interval) Useful for abstract rhythms.
<b>OUTLET 4:</b>	Bang on pattern complete. Useful for advancing to next pattern/sequence or for chaining Lattices together.

## UI Overview:



## Lattice header:

- Displays running UI statistics vital to the user:

Lattice Name		mute on/off
Pattern #	Row #	lock on/off
Octave	Reference Frequency	mono mode on/off

## Lattice row (value):

- Stores FLOAT,INT,MESSAGE,LIST, Symbol & META commands

36.333302 / 326 / hello / 1,2,3 / @rep / @del / ;sendms bling

## Lattice row (indicator):

- Indicates the active row being written/edited/played.
- Displays selected regions for cut/copy/paste commands
- Indicates the Major and Minor row intervals (for time signature use – values output at index outlet)
- Indicates muted rows by displaying a circle

circle indicates muted row	O

## Summary of editing lattice row content:

By striking shift-right-arrow or shift-left-arrow the user can open a lattice row for regular editing similar to a text box. The following keys will help you:

Enter	End editing, send (interpreted) row contents through left outlet.
Tab	End editing.
Return	End editing, send (interpreted) row contents through left outlet, select next row (previous row if shift key is depressed).
Up Arrow	End editing, go to previous row, automatically re-enter edit mode, maintaining the same text selection as was previously the case. If the option key is depressed, then edit mode will not be re-entered.
Down Arrow	End editing, go to next row, automatically re-enter edit mode, maintaining the same text selection as was previously the case. If the option key is depressed, then edit mode will not be re-entered.
Clear	Currently selected characters will replace by spaces (Note that, in accordance with Max conventions, multiple



spaces are compacted to a single space when storing data.)

Backspace	Delete selected characters, move cursor to character before the first selected character. If the selection started at the first character, then after the deletion the first character of the remaining text will be selected.
Left Arrow	Select previous character
Right Arrow	Select next character
Delete	Delete current selection; the selection cursor remains at the position of the first selected character prior to deletion.

~~ anything else will be entered as text into the row ~~

## Navigating Lattice:

- First keystrokes are examined to see if they occur in the current charmap. If so, they are interpreted according to the charmap. Otherwise:



Home	Select first row in current pattern (scroll if necessary to make it visible) If control key is depressed, select first row of first pattern.
Enter	Same as bang message.
Control-C	Copy selection to lattice clipboard.
End	Select last row in current pattern (scroll if necessary to make it visible).
Control-E	Exponential interpolation between first and last values of current selection (Same as 'exp 2.71828' message)
Control-F	Convert all selected rows containing integers to floating point format.
Backspace	Deletes row and moves all row data beneath deleted row up 1.
Tab	Enter edit mode for first row in selection
Control-I	Convert all selected rows containing floating-point values to integers (truncating anything after the decimal point)
PageUp	Select row N rows before the first currently selected row, where N is the Minor Emphasis Interval. If the option key is depressed, N is the Major Emphasis Interval. If the shift key is depressed, the selection is extended to include all rows from the newly selected row to the last row of the previous selection.
PageDown	Select row N rows after the last currently selected row, where N is the Minor Emphasis Interval. If the option key is depressed, N is the Major Emphasis Interval. If the shift key is depressed, the selection is



	extended to include all rows from beginning of the previous selection to the newly selected row.
Return	Send (interpreted) row contents through left outlet and select next row (previous row if shift key is depressed).
Control-R	Convert all selected rows containing floating-point values to integers by rounding.
Clear	Clear all rows in current selection
Control-V	Paste contents of lattice clipboard, starting at the first row of the current selection.
Control-X	Cut contents of current selection to lattice clipboard.
LeftArrow +option:	transpose all values in current selection down by one half-step. The exact value of the "half-step" may depend upon the micro-tuning settings. Otherwise: begin editing contents of current row.
RightArrow +option:	transpose all values in current selection down by one half-step. The exact value of the "half-step" depends upon the micro-tuning settings. Otherwise: begin editing contents of current row.
UpArrow:	select the row before the first row in the current selection.
+control:	make previous pattern active.
+shift:	add the row before the first selected row to the current selection.
+shift +option:	remove the first selected row from the current selection.
+option:	"nudge" up all values in the currently selected rows. "Nudging" means to add one to all integer values; with floats the third significant digit is incremented.
DownArrow:	select the row after the last row in the current selection.
+control:	make next pattern active.
+shift:	add the row after the last selected row to the current selection.
+shift +option:	remove the last selected row from the current selection.
+option:	"nudge" down all values in the currently selected rows. "Nudging" means to subtract one to all integer values; with floats the third significant digit is decremented.
Space	Repeat last message buffered by a fin or nof symbol and go to next row
Control-Space	Store fin symbol and go to next row
Shift-Control-Space	Store nof symbol and go to next row
Control-Plus	Transpose all values in the selected rows up by one octave. The exact value of the "octave" may depend upon the micro-tuning settings. Note: This is the Plus key on the number pad, or for laptops hit fn+ctrl+?
Control-Comma	Set reference frequency for charmap shortcuts down one octave. The exact value of the "octave" may depend upon the micro-tuning settings.

Control-Minus	Transpose all values in the selected rows down by one octave. The exact value of the "octave" may depend upon the micro-tuning settings. Note: This is the Minus key on the number pad, or for laptops hit fn+ctrl+;
Period	Delete contents of currently selected rows and move selection to the row after the last row of the current selection.
Control-Period	Set reference frequency for charmap shortcuts up one octave. The exact value of the "octave" may depend upon the micro-tuning settings.
Delete	Delete the current row, shifting following rows up by one With either of the shift or control keys (or even both!) a new empty row will be inserted instead. Note: This is the Del key on the number pad, or for laptops hit fn+ctrl+Delete

## In Row Meta Commands

@rep	(repeat x,y,z)
@del	(delay message x ms)
nof	(note off (drop second inlet 0))
fin	(finish note and echo note off message in MIDI friendly manner).



## Input – Left Inlet

**bang**     Synonym for the 'next' message.

**next**     The contents of the current row are sent through the left outlet and the playback cursor advances to the next row. This is normally the row immediately beneath the current row, but see the nav message for information about the different playback navigation modes supported by lattice.

When lattice reaches the last row in the current pattern, it automatically continues with the first row of the pattern on receiving the next message.

If the row contains any data, the value 1 is sent through the second outlet to indicate that the object is "active". See the fin and finecho messages for more information.

Additionally, the current row number is sent through the third outlet. A bang is sent through the rightmost outlet when the last row of the pattern has been banged.

Right-to-left order in processing outlets is observed.

**prev**     The contents of the current row are sent through the left outlet and the playback cursor advances to the previous row. This is normally the row immediately above the current row, but see the nav message for information about the different playback navigation modes supported by lattice.

When lattice reaches the first row in the current pattern, it automatically continues with the last row of the pattern on receiving the prev message

int The integer value is stored in the current lattice row.

Lattice supports a variety of different formats for displaying integers. Please refer to the Inspector section, below.

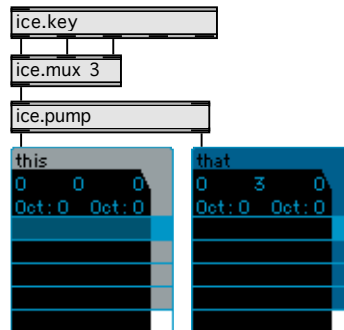
float The floating-point value is stored in the current lattice row.

Lattice supports a variety of different formats for displaying floats. Please refer to the Inspector section, below

list The list is stored in the current lattice row.

char The word char followed by three integers triggers a keyboard short cut. See the section on Keyboard Shortcuts below for a complete list of supported keyboard commands. The three parameters to the char message are typically taken from the left three outlets of an ice.key object (or the three outlets of the stock key object).

The ice.pump object automatically prepends the char message to three-item integer lists, so a configuration such as the following is the idiomatic way of passing keystrokes into ice.lattice. (In this example, the leftmost lattice currently has focus from the ice.pump object; it would also be possible to use a pack object with exactly integer three inlets instead of the ice.mux object.)



charmap

As an alternative or complement to the factory-installed shortcuts described in the section Keyboard Control below, you can define your own keyboard shortcuts. The symbol charmap followed by two or more parameters defines custom keyboard shortcuts. The first parameter must be an int in the range 0-255, representing the ASCII code generated by the keyboard shortcut you wish to define. The subsequent parameters can be ints, floats, or symbols. The second parameter will then be entered into the current row of the lattice whenever a char message is received that contains the ASCII code you defined. Subsequent ints, floats, and symbols will be mapped to the following ASCII codes. For instance, the following message:

charmap 49 do re mi fa sol la ti

Would cause the symbol 'do' to be entered whenever you press the key 1



(ASCII code 49); pressing the key 2 (ASCII code 50) would enter the symbol 're'; pressing the key 3 would enter the symbol 'mi', and so on. As an other example, consider

```
charmap 65 440.0 495.0 594.0 586.66667 660.0 704.0 782.22222
```

This would map the uppercase letters A-G to frequencies forming an Aeolian scale in just intonation.

For convenience in working with some popular tunings three sets of charmaps are predefined and can be called up with the symbols charmap eqtemp (for equal-tempered scales), charmap lucy (for the Lucy scale, a 28 pitch-per-octave scale based on an extended circle of fifths), and charmap mididata (for entering MIDI note numbers). The keyboard mappings are displayed in the section Predefined Charmaps, below.

ASCII codes defined by charmap message override all factory-defined keyboard shortcuts with the same ASCII code. There are sometimes several different keystrokes producing the same ASCII code. The factory-installed keyboard shortcuts can differentiate between these keystrokes by examining the hardware keycode (the second outlet from ice.key or key) and the modifier keys, such as Option or Control, that were depressed (these are encoded in the integer sent through the third outlet from ice.key or key). If you find that your charmap definitions interfere with factory-defined keyboard shortcuts, you may need to refer to a technical documentation of keyboard layouts to resolve the issue.



go

The symbol go followed by a positive integer moves the playback cursor down by the number of rows specified. If followed by a negative integer, the playback cursor is move upwards by that number of rows.

Several keystroke shortcuts (page up, page down, etc.) are predefined for moving the cursor in steps of four and sixteen rows. Please see the Keyboard Control section for details.

goto

The symbol goto followed by an integer moves the playback cursor to the row specified. The integer parameter is clipped to the size of the current pattern. Row numbering begins with row zero.

Several keystroke shortcuts (home, end, etc.) are predefined for moving the cursor in to the beginning and end of the pattern. Please see the Keyboard Control section for details.

selstart

The symbol selstart followed by a single integer extends or contracts a selection of rows. It specifies a new row for the selection to start at without modifying the end of the selection. This message is typically triggered by keyboard shortcuts.

selend

The symbol selend followed by a single integer extends or contracts a selection of rows. It specifies a new row for the selection to end at without modifying the beginning of the selection. This message is typically triggered by keyboard shortcuts.

select

The symbol select followed by two integers specifies a range of selected rows. This message is typically triggered by keyboard shortcuts.



editset

Used by ice.pump to coordinate the current selection of multiple synchronized ice.lattice objects.

patgo

The symbol patgo followed an integer changes the currently active pattern. The current playback and editing row selections normally do not change position due to a patgo message. However, in the case where new active pattern is smaller than the previously active pattern, playback and editing row selections are constrained to the dimensions of the new active pattern.

Note: Consistency within the ice.lattice navigation model would demand that this message be named 'patgoto'. However, in the course of developing ice.lattice it was felt that there was little need for two different messages for navigating between patterns and that a direct 'goto'-like message was the more functional behavior. Since shorter message names were preferred to long names, the function was dubbed patgo. There is now a sufficient body of work using this message name that the effort in changing numerous existing patches would outweigh the advantage of more consistent naming conventions.

seek

The symbol seek followed by two integers changes the pattern and row in one handy message. The first integer specifies the new active pattern, the second integer specifies the new playback row.

nav

The symbol nav followed by an integer specifies a navigation mode.

The integer specifies the navigation type. Currently supported values are: 0 (forward, the default setting), 1 (backward), 2 (bounce). In forward mode, the bang message moves the playback cursor sequentially from top to bottom (in ascending row number order). When the last row is reached, navigation continues from the top of the pattern. In backward mode, the bang message moves the playback cursor in the opposite direction, from bottom to top (in descending row number order). When the first row is reached, navigation continues from the bottom of the pattern. In bounce mode, the bang message moves the playback cursor from top to bottom until it reaches the last row, then direction is reversed. When the playback cursor reaches the top row, direction is reversed again. In all cases the next message behaves identically to bang, and the prev message navigates in the opposite direction from bang.

pumpstate

This message is used by ice.pump to specify which of the connected ice.lattice objects will respond to keyboard commands.

octgo

octgoto

By default, when entering MIDI note numbers and frequencies with keyboard shortcuts, ice.lattice interprets keystrokes relative to Middle C. This is called the reference octave, and Middle C is the reference octave zero. The octgo and octgoto messages allow you to modify the reference octave.

The symbol octgo followed by a positive value will transpose keyboard shortcuts up by that many octaves, negative values transpose the keyboard shortcut values down. Octgo messages are cumulative: 'octgo 2' followed by 'octgo -4' will result in MIDI keyboard shortcuts being transposed down two octaves from the starting octave range.





- The symbol `octgoto` followed by an integer will cause keyboard shortcuts used afterwards to be calculated relative to the octave specified.
- octref**
- By default, all frequencies are calculated relative to Middle C = 261.6255653006 Hz. This is equivalent to A = 440 Hz at equal temperament. However, not everyone in the world tunes to A 440. The symbol `octref` followed by a floating point value allows you to specify a different reference frequency for Middle C. For instance, to prepare a piece to be played with the Berlin Philharmonic, you might want to start by sending the message `octref 265.78779` to `ice.lattice`.
- octrel**
- By default the octave is defined as a frequency ratio of 2:1. However, if you want to work with exotic intonations, such as those used in Karlheinz Stockhausen's seminal electroacoustic composition "Studie II", you can use the symbol `octrel`, followed by a float, which will specify an alternative "octave" ratio. For instance, Studie II could be realized using `octrel 2.236068`
- octdiv**
- By default the octave is divided into 12 equal steps. However, if you want to work with exotic intonations, such as those used in Karlheinz Stockhausen's seminal electroacoustic composition "Studie II", you can use the symbol `octdiv`, followed by an int, which will specify an alternative subdivision of the octave. For instance, Studie II could be realized using `octdiv 5`
- trans**
- The symbol `trans` followed a number, will transpose the frequencies in the currently selected rows by the number of scale steps specified. Negative number transpose downwards. The number may be a float, in which case a microtonal interval will be applied.
- The `trans` message is only applied to rows with numeric content (simple ints or floats, not lists or other messages). In synchronized editing mode, only the playback row is affected by this message.
- The `trans` message respects the current `octrel` and `octdiv` settings.
- oct**
- The symbol `oct` followed an int, will transpose the frequencies in the currently selected rows by the number of octaves specified. Negative number transpose downwards.
- The `trans` message is only applied to rows with numeric content (simple ints or floats, not lists or other messages). In synchronized editing mode, only the playback row is affected by this message.
- The `oct` message respects the current `octrel` setting.
- tune**
- The `tune` message is used to define arbitrary, possibly non-equal tempered, tuning schemes. The symbol `tune` is followed by a list of floats. The list will typically contain 11 elements (or one less than whatever the current `octdiv` setting indicates). The first element of the list specifies a frequency ratio between the first and second step of your tuning table; the second element specifies a frequency ratio between the second and third step; and so on. If you specify a list that is longer than needed, it will be truncated. If you specify fewer intervals than the scale has, the highest intervals will simply remain unchanged from the previous setting (which was probably equal-tempering).



## detune

The detune message functions similarly to the tune message, but instead each item in the list specifies an amount to detune the respective step of the scale from equal temper. Each value of 0.01 represents a detuning of one cent (sharp). Use negative values to lower pitch.

## step

The symbol step followed by an integer sets all selected rows to the scale step specified.

The step message replaces the contents of all selected rows, regardless of whether they held numeric or other information beforehand. In synchronized editing mode, only the playback row is affected by this message.

The step message respects the current octrel and octdiv settings.

## nudge

The symbol nudge followed by an integer “nudges” all simple numeric values in the currently selected rows up or down by a certain amount. Rows containing lists, symbols, or other messages are unaffected. This is typically used to fine-tune values already entered.

For integers “nudging” simply means to increment the current value by the amount specified. Negative nudge values decrement the existing value.

For floats, nudging increments (or decrements) the third significant digit by the specified amount. For example, the message ‘nudge 2’ would change a row containing the value 9.01 to 9.03; the value 10.1 would become 10.3, and the value 0.05 would become 0.0502.

## focus

The focus message is used by ice.pump to tell connected ice.lattice objects to draw themselves to reflect whether pump is giving the object focus. This message will not typically be used by the user.

## del

The del message clears all currently selected rows.

## clear

The symbol clear followed by an integer deletes the contents of entire patterns. If the integer is zero, the currently active pattern is cleared. If the integer is a positive value, the pattern with that number will be cleared. If the integer is -1, then all patterns will be cleared. Use carefully.

## edit

The message edit causes ice.lattice to enter row content editing mode. You can edit row data with a text editor reminiscent of old-fashioned tracker editing conventions. Please consult the section on Row Editing Conventions below for further details.

## lin

The lin message causes ice.lattice to fill all selected rows with numeric values, linearly interpolating between the values in the first and last rows of the selection. This is, for instance, useful in generating MIDI Continuous Controller data.

The selection must encompass at least three rows for this to have any effect.

## pow

Like the lin message, pow is used to fill the currently selected rows with values interpolated between the first and last rows. However, pow generates curved interpolations. These curves are often incorrectly referred to as

exponential (see exp below for true exponential interpolation).

The symbol pow followed by a positive float fills the selected rows with values interpolated between the values currently stored in the first and last rows of the selection. The float parameter determines how steep the curve is. Values close to one will give an almost linear curve; larger values will give a curve with small steps at first, then larger steps; values closer to zero will give a curve with first large steps, then smaller steps as you get closer to the final value.

The shape of the curve is determined by the formula  $y = x^p$ , where p is the parameter to the pow message. The value of x is calculated by ice.lattice such that the resulting values of y run from the value of the first row in the current selection to the value of the last row in the selection.

exp

The symbol exp followed by a positive float interpolates between the first and last values of the current selection using the formula  $y = x^p$ , where x is the parameter to the exp message and p is calculated by ice.lattice according to the values found in the first and last rows of the current selection. The values generated are slightly different from those generated by the pow message.

type

The type message is used to force all numeric values in the currently selected rows to become either all float or all int.

If the symbol type is followed by a float parameter (and we don't care about the value—any float will do), then all selected rows with numeric content are converted to floats.

If the symbol type is followed by an int parameter, all selected rows with numeric content are converted to int, and the value of the parameter determines the type of rounding used. The following values are supported:

- 0 truncate: the fractional portion is simply truncated. This is what Max normally does when converting floats to int. Although this is documented behavior, it continues to surprise many people.
- 1 round: the schoolbook rounding most people want: if the fractional part is greater than or equal to 0.5, round upwards, otherwise round downward (for negative values this logic is reversed).
- 2 floor: the largest integer less than or equal to the original float value. For positive values this is the same as truncation but for negative values the result is different.
- 3 ceiling: the smallest integer greater than or equal to the original float
- 4 to infinity: positive values are rounded upwards to the next integer, negative values are rounded down.

Float:	1,0	1,1	1,5	1,99	-1,0	-1,5	-1,9
--------	-----	-----	-----	------	------	------	------



type 0	1	1	1	1	-1	-1	-1
type 1	1	1	2	2	-1	-2	-2
type 2	1	1	1	1	-1	-2	-2
type 3	1	2	2	2	-1	-1	-1
type 4	1	2	2	2	-1	-2	-2

mono  
lock  
snap  
sync  
enterbang

These messages are “mode” messages, controlling modes of operation in an ice.lattice object. They all take a single integer parameter. A value of zero turns the mode off; a value of one turns the mode on; and a value of -1 toggles the state of the mode. If no argument is provided, these messages turn the mode of operation *on*. (This is the opposite of the way many other mode-type messages in Max function, but it seems much more logical for a message such as ‘lock’, without parameters, to turn lock mode on rather than turn lock mode off.)

When mono mode is on, ice.lattice buffers the last message sent through the left outlet and automatically resends it the when the next row with content receives a bang, next, or prev message. This mechanism is useful for implementing a conventional mono sequencer.

When lock mode is on, row contents cannot be modified. All editing messages are disabled.

When snap mode is on, navigation commands that jump ‘beat-by-beat’ or ‘bar-by-bar’ jump to the next minor division (“beat”) or major division (“bar line”), even if the playback cursor was between division markings when the command is received.

In sync edit mode, the input row is synced to the playback row. As a consequence you cannot have a multi-row selection (for messages such as lin, pow, type, etc.) in sync edit.

When enterbang mode is on, char messages that represent the Enter key (typically char 3 76 0 or char 3 52 0) are interpreted as synonyms for bang.

The modes mono and lock can also be toggled by clicking in the header of an ice.lattice object (see “Lattice Header and Mouse Actions” below).

mute

The mute message is a mode message, like mono, lock, etc. Use it to mute individual rows or an entire lattice. The symbol mute can be followed by up to two integer arguments. The first integer specifies the row to turn muting on or off for. If the row number is -1, then the muting is applied to the entire ice.lattice. When the second integer is 1, muting is turned on; if zero, muting is turned off; -1 toggles the mute state.

Mute mode can be toggled for individual rows by clicking in the square at the right of the row; the entire ice.lattice object can be muted or unmuted by clicking in the header (see “Lattice Header and Mouse Actions” below).

setname  
setcolor  
setemph  
setpatcount  
setpatlen  
size

These messages are used the ice.lattice Inspector to control certain behaviors. You can also use these messages yourself.

The symbol setname followed by a symbol sets the object’s name to the message argument.



setintformat  
setfloat-  
format

The symbol setcolor, followed four integers, modifies the object's color scheme. The second, third, and fourth values must be in the range 0 – 255 and specify amounts of red, green, and blue in the color. The first integer specifies which of the following colors to modify:

- 0 text
- 1 title text
- 2 title background
- 3 background color to use when the object has focus
- 4 background color to use when the object is in bang mode (ie, bang and similar messages passed through an ice.pump object are communicated to all connected instances of ice.lattice)
- 5 background color to use when the object is in navigation mode (ie, navigation commands passed through an ice.pump object are used to synchronize multiple instances of ice.lattice).
- 6 row background color
- 7 color used to indicate rows at the major emphasis interval ("bar lines")
- 8 color used to indicate rows at the minor emphasis interval ("beats")
- 9 color used to indicate the row selection cursor

The symbol setemph followed by two positive integers sets the emphasis intervals. The first number sets the minor emphasis interval ("beat") and the second number sets the major emphasis interval ("bar length"). These settings are primarily cosmetic, there to help you keep track of where you are in your sequence. They have no effect on how ice.lattice interprets the contents of the rows. However, navigation commands triggered by Page Down, Page Up, and similar keys abide by your emphasis settings. The default values are 16 rows/major emphasis and 4 row/minor emphasis. The smallest valid major emphasis setting is 4 rows and minor emphasis is always constrained to be less than the major emphasis value. The largest major emphasis setting is a ridiculously large number, it is highly unlikely that you would ever want to go that high.

The symbol setpatcount followed by a positive integer sets the number of patterns maintained by the object. By default each ice.lattice has 16 patterns, but you can have as few as a single pattern and up to 16,000 patterns.

The symbol setpatlen followed by a positive integer sets the length of the current pattern. By default each pattern has 128 rows, but you can have patterns with as few as a single row or as many as 2,147,483,647 rows.

Note that changing the pattern length is currently a *destructive* edit. If you shorten the pattern length, any data stored in the rows beyond the new last row are gone. This is like removing circuitry from your hardware sequencer. Be careful.

The size message is a synonym for setpatlen.

The setintformat and setfloatformat messages determine how ice.lattice formats numbers. Either message takes a single integer parameter.



These are the integer formats ice.lattice understands

- 0 decimal (default setting)
- 1 hexadecimal
- 2 Roland Octal
- 3 Binary
- 4 MIDI note numbers (Middle C, 60, defined as C3)
- 5 MIDI note numbers (Middle C, 60, defined as C4)

And these are the float formats:

- 0 decimal (default setting)
- 1 display as nearest MIDI note number for the frequency specified. Middle C, 60, defined as C3
- 2 display as MIDI note number for the frequency specified, display detuning from equal temperment as a decimal fraction (0.01 is the equivalent of one cent). Middle C, 60, defined as C3.
- 3 display as nearest MIDI note number for the frequency specified Middle C, 60, defined as C4
- 4 display as MIDI note number for the frequency specified, display detuning from equal temperment as a decimal fraction (0.01 is the equivalent of one cent). Middle C, 60, defined as C4
- 5 A generalized note-naming scheme for “scales” that deviate from the 12-note-per-octave tradition (cf. the octdiv and octrel messages above). Frequencies displayed as the closest matching note name.
- 6 The same generalized note-naming scheme, with deviations from the named notes indicated as decimal fractions (0.01 is equivalent to 1/100 of the interval between two adjacent steps).

tattle

Displays a summary of information about the object in the Max window.

vers

The symbol vers followed by two ints and an optional symbol, generate version information for the object class. The symbol, if specified, causes the version information to be sent to a named receive object. Otherwise the version information is sent through the left outlet. In this case the vers message would typically be used with a grab object.

The two integers specify the version ID and version type, respectively. The version ID can be either one (referring to the object class) or two (referring to the iCE Tools package as a whole). The version type can be one (a brief text representation), two (a longer text representation), or three (a binary numeric representation of the version number).

read

The read message causes ice.lattice to read its contents from a document on disk. You can name a file by including a symbol argument to the read message. If no name is specified, a standard Select File dialog (or Property Sheet) will be displayed in which you can select an ice.lattice document to read.

write

The write message causes ice.lattice to store its contents in a document on disk. You can name the file by including a symbol argument to the write message. If no name is specified, a standard Save As... dialog (or property Sheet) will be displayed in which you can specify where to store the





	ice.lattice document.
readagain	Rereads an ice.lattice document previously specified by a read message. Any modifications made since the last read command are overwritten. Use with care. If no file has been previously specified, the message behaves like a simple read message, going through the “Select File” rigmarole.
writeagain	Saves the current contents of the ice.lattice object to the file last specified by a write message. There are no warnings about overwriting existing files, use with care. If no file was previously specified, this behaves like a simple write message, Save As... dialog and all.
ice.pump. master	This message is reserved for communication between ice.pump and ice.lattice objects. It is strongly recommended not to use this message for other purposes.
store	Store the arguments to the message in the current row. You only need to use this if the message you want to store is one of the above-named messages.
@rep	The symbol @rep is followed by a positive float, a positive int, and up to 253 additional parameters. The message, in its entirety, is stored in the current row. When the row is banged, instead of just sending the row contents out the left inlet once, the row output is repeated, at time intervals specified by the float argument, for the number of repetitions specified by the int argument. The time interval between is given in <i>Hertz</i> (repetitions per second). The symbol @rep, the time interval, and the repeat count are all stripped from the message and only the remaining arguments are sent through the left outlet.
@del	<p>The symbol @del is followed by a float and up to 254 additional parameters. The message, in its entirety, is stored in the current row. When the row is banged, instead of immediately sending the row contents out the left inlet, the row data are delayed by an amount of time specified by the first argument. The delay time is given as an interval in <i>Hertz</i> (cycles per second) for consistency with the @rep message.</p> <p>It is legal to concatenate @del and @rep messages. For instance, my wife might want to program the following:</p> <p style="text-align: center;">@del 0.001 @rep 0.01 999 Take out the rubbish!</p> <p>to remind me about household duties.</p> <p>Note, however, that a maximum of 256 items that can be stored in any row message, so concatenating @del and @rep messages reduces the number of additional items you can store in the row. This is only an issue in exceptional circumstances.</p>
fin	The symbol fin is stored in the row. When a row containing this message is banged, it is treated as a kind of generic “note-off” message, sending a zero out the second outlet.
finecho	Like the fin message, but in addition when the row is banged the last message sent through the left outlet is repeated. This can be convenient for



anything generating MIDI note-off messages.

Anything else is stored in the current row. Really, anything. You can even connect MSP and Jitter patch cords to ice.lattice, although this will probably not do what you want (unless you have an exceptionally thorough technical understanding of how these objects communicate). Ints, floats, lists, complex messages are all stored as one would expect, including all punctuation conventions used in standard Max message boxes. Row contents will be sent through the left outlet whenever a bang, next, or prev message is received in the left inlet.

## Font Menu

All ice.lattice objects respond to the Max Font menu to specify a font family and font size.

## Input – Middle Inlet

anything The message is stored in the current row. No message interpretation occurs.

## Input – Right Inlet

int The equivalent of a patgo message in the left inlet.

## Output

all manner of stuff The rightmost outlet: a bang is sent whenever the last row in the current pattern is banged. Which row is the “last” depends upon navigation settings, but this is typically the one at the very bottom.

Third outlet: the row number of the current row is sent whenever a bang, next, or prev message is received in the left inlet.

Second-outlet: one is sent whenever a row containing data is banged. Zero is sent when a row with the fin or finecho message is banged.

Left outlet: the contents of the current row are sent through this outlet whenever a bang, next, or prev message is received in the left inlet. Note, however, the special handling of the messages @rep, @del, fin, and finecho described above.





## Inspector

iCE Lattice object by Anthony Bisset & Peter Castine  
Copyright DSPaudio, Inc. 1999-2006

Version 1.0.4

Lattice  
Name  
Colors

Set the name of the object (displayed in the Lattice Header).

Select the item that you want to color from the popup menu. Choose from Text, Title Text, Standard Title Background, Focus Title Background, Title Background/Bang mode, Title Background/Navigation Mode, Row Background, Major Emphasis Indicator, Minor Emphasis Indicator, and Cursor Indicator. Specify the color either using the Max swatch object or the more standard Color Picker

Emphasis  
Intervals

Set major and minor emphasis intervals (“bars” and “beats”). These settings are primarily cosmetic, there to help you keep track of where you are in your sequence. They have no effect on how ice.lattice interprets the contents of the rows. However, navigation commands triggered by Page Down, Page Up, and similar keys abide by your emphasis settings. The default values are 16 rows/major emphasis and 4 row/minor emphasis. The smallest valid major emphasis setting is 4 rows and minor emphasis is always constrained to be less than the major emphasis value. The largest major emphasis setting is a ridiculously large number, it is highly unlikely that you would ever want to go that high.

Number of  
Patterns

Sets the number of patterns contained by the object



## Pattern Length

Sets the length of all patterns in the lattice object. Currently if you want to have patterns of different length in a lattice object, you must send setpatlen messages to each pattern, one at a time. Also recall that shortening the pattern length is a destructive operation (data in rows no longer used is *not* buffered—think of this as physically removing circuitry from a hardware sequencer).

To reduce the opportunity for accidentally changing pattern length, you need to first unlock the number box by toggling the Lock Pattern Length checkbox off.

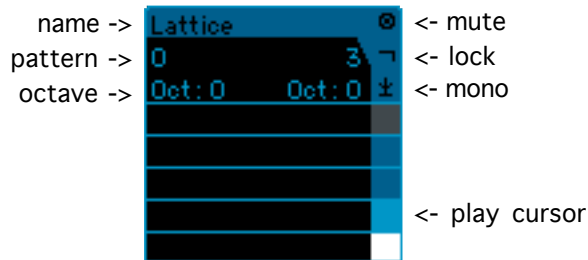
## Integer Format

## Float Format

Select an integer format from the popup menu.

Select a float format from the popup menu.

## Lattice Header and Mouse Actions



The Lattice name, displayed left justified at the top of the object, is for display purposes.

The second line of the header (colored with the normal row background) displays the pattern number (at the left) and current row (at the right). If async row editing is enabled, the number at the right is the playback row and the row selection range is displayed in the middle.

The third line of the header displays the current octave at the left and frequency of the reference octave at the right.

Clicking on a lattice object gives it focus. Lattice objects communicate with an ice.pump object to maintain focus and to route keyboard commands to those Lattice objects that currently have focus.

A Lattice that has focus responds to mouse clicks as follows:

- The three icons for mute, lock, and mono can be clicked to toggle the state.
- You can click in the box at the right of any row to toggle mute state for that single row.
- Clicking on a row makes that row the current row.

## Keyboard Control

Keyboard control is typically mediated by either the stock key object or by ice.key, as shown above in the description of the char command. The ice.pump object will automatically prepend the symbol char to incoming integer triples. For controlling a single instance of ice.lattice, the following configuration might be used:





Keystroke	Message Equivalent	Action
Home	goto 0	Move Playback cursor to first row of current pattern
Ctrl-Home	seek 0 0	Make Pattern 0 active and move playback cursor to first row of pattern.
End	goto 214748367	Move Playback cursor to last row of current pattern
PgUp		Move Playback cursor up by minor emphasis interval
Shift-PgUp		Extend selection, moving selection start up by minor emphasis interval
Option-PgUp		Move Playback cursor up by major emphasis interval
Shift-Option-PgUp		Extend selection, moving selection start up by major emphasis interval
PgDown		Move Playback cursor down by minor emphasis interval
Shift-PgDown		Extend selection, moving selection start down by minor emphasis interval
Option-PgDown		Move Playback cursor down by major emphasis interval
Shift-Option-PgDown		Extend selection, moving selection start down by major emphasis interval
Left Arrow		Switch to row edit mode
Option-Left Arrow	trans -1.0	Transpose down by (half) step
Right Arrow		Switch to row edit mode
Option-Right Arrow	trans 1.-	Transpose up by (half) step
Up Arrow		Move Playback cursor up one row (taking navigation mode into account)
Shift-Up Arrow		Extend selection, moving selection start up one row
Option-Shift-Up Arrow		Contract selection, moving selection start down one row
Ctrl-Up Arrow		Make previous Pattern active
Down Arrow		Move Playback cursor down one row (taking navigation mode into account)
Shift-Down Arrow		Extend selection, moving selection start down one row
Option-Shift-Down Arrow		Contract selection, moving selection start up one row



Ctrl-Down Arrow		Make next Pattern active
Enter	bang	Next (in enterbang mode only)
Return	next	Bang (in enterbang mode only)
Shift-Return	prev	Prev (in enterbang mode only)
FwdDel	del	Clear contents of selected rows
Delete		Delete one row from pattern
Ctrl-Shift-Delete		Insert one empty row into pattern
Tab		Switch to row edit mode
Space		Repeat last banged message
Ctrl-Space		Enter fin message into current row
Ctrl-Shift-Space		Enter finecho message into current row
. (Period)		Clear contents of current row and move playback cursor to next row
Ctrl-A	sel 0 214748367	Select All (only available in async row editing)
Ctrl-E	exp 1	Simple exponential interpolation between first and last selected row (only available in async row editing)
Ctrl-F	type 1.0	Change type of all selected integers to float
Ctrl-I	type 0	Truncate all selected floats to int
Ctrl-P	pow 2	Parabolic interpolation between first and last selected row (only available in async row editing)
Ctrl-Q	sync -1	Toggle sync mode
Ctrl-R	type 1	Round all selected floats to int
Ctrl-+		Transpose up an octave
Ctrl-—		Transpose down an octave
Ctrl-, Ctrl-<	octgo -1	Transpose reference octave for MIDI data entry down
Ctrl-> Ctrl-.	octgo 1	Transpose reference octave for MIDI data entry up

## Predefined Charmaps

The following charmaps are predefined.

charmap      Enter frequencies for Lucy circle-of-fifths scale.  
lucy

c# 1	d# 2	e# 3	f# 4	g# 5	a# 6	b# 7
c Q	d W	e E	f R	g T	a Y	b U
c ♭ A	d ♭ S	e ♭ D	f ♭ F	g ♭ G	a ♭ H	b ♭ J

c ♭ ♭	d ♭ ♭	e ♭ ♭	f ♭ ♭	g ♭ ♭	a ♭ ♭	b ♭ ♭
Z	X	C	V	B	N	M

With the shift key the frequency is automatically entered one octave higher.

charmap  
eqtemp Enters frequencies for equal tempered scale.

1	c # "	d # "	4	f # "	g # "	b ♭ "
2	3		5	6	7	
c "	d "	e "	f "	g "	a "	b "
Q	W	E	R	T	Y	U
	c # '	d # '	F	f # '	g # '	b ♭ '
A	D		G	H	J	
c '	d '	e '	f '	g '	a '	b '
Z	X	C	V	B	N	M

With the shift key the upper two ranks are entered one octave higher; the lower two ranks are entered one octave lower.

charmap  
mididata Enters MIDI note numbers.

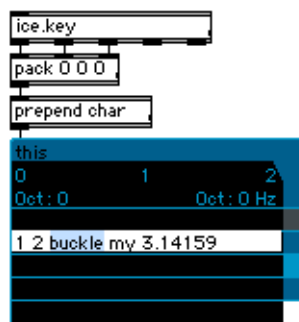
1	61	63	4	66	68	70
2	3		5	6	7	
60	62	64	65	67	69	71
Q	W	E	R	T	Y	U
	49	51	F	54	56	58
A	S	D	G	H	J	
48	50	52	53	55	57	59
Z	X	C	V	B	N	M

With the shift key the upper two ranks are entered one octave higher; the lower two ranks are entered one octave lower.

## Row Editing

The message char 7 48 0 (the tab key) causes ice.lattice to enter row editing mode, in which you can modify the contents of the currently selected row character-by-character.





Row editing resembles traditional tracker editing conventions more closely than the standard Mac OS TextEdit conventions, but there are certain similarities between the two. You may prefer to use a monospace font to enhance the tracker look-and-feel.

The main difference to Mac OS TextEdit is that the smallest selection unit is a single character, which will be displayed with a block cursor. The cursor color is whatever you have selected as your global highlighting color. The other significant difference is that editing is always in overwrite mode, with the exception of the space key, which inserts space.



Keystroke	Action
Enter	Exit Row Edit mode, send new contents through left outlet; move to next row.
Tab	Enter Row Edit mode.
Return	Exit Row Edit mode, send new contents through left outlet; move to next row.
Shift-Return	Exit Row Edit mode, send new contents through left outlet; move to previous row.
Arrow Up	Move to previous row, remain in Row Edit mode
Option- Up Arrow	Move to previous row, remain in Row Edit mode, maintaining the character selection
Down Arrow	Move to next row, remain in Row Edit mode.
Option-Down Arrow	Move to next row, remain in Row Edit mode, maintaining the character selection.
Shift-PgDown	Extend selection, moving selection start down by minor emphasis interval
Option-PgDown	Move Playback cursor down by major emphasis interval
Clear	Delete current selection
Backspace	Delete character to the left of the cursor
Delete	Delete (one) character under the cursor, moving any characters to the right of the cursor leftwards
Left Arrow	Move cursor one character to the left

Shift-Left Arrow	Extend selection by one character to the left
Right Arrow	Move cursor one character to the right
Shift-Right Arrow	Extend selection by one character to the right

## ICE.PUMP Message Summary:

This being the complete List of Messages, that the Object so-named "ice.pump" doth understand (and exposeth unto sundry Users); herewith following partial Documentation of what said Messages do (if and sobeit received by way of Inlet on the Side closer to the Left Hand):

bang	same as 'next'
int	send value through currently active outlet (or all outlets if in a Spray Mode)
float	send value through currently active outlet (or all outlets if in a Spray Mode).
list	Lists containing exactly three integers have a special meaning: they are interpreted as information defining a keystroke (ASCII code, keyboard scan code, modifier keys). If a list has this form <code>_and_</code> the information represents one of the keystrokes with special meaning for ice.pump (see below/above/wherever) <code>_and_</code> the list arrives in the left inlet, then the list is interpreted by the ice.pump object. Otherwise the list is passed out through the currently active outlet (or all outlets if in a Spray Mode).
prev	in left inlet: deactivate the current outlet and activate the next outlet to the <code>_left_</code> . If the currently active outlet is the leftmost outlet, cycle around to the rightmost outlet. in right inlet: simply pass message through current outlet (or all outlets if in a Spray Mode) without interpretation.
next	in left inlet: deactivate the current outlet and activate the next outlet to the <code>_right_</code> . If the currently active outlet is the rightmost outlet, cycle around to the leftmost outlet. in right inlet: simply pass message through current outlet (or all outlets if in a Spray Mode) without interpretation.
goto <int>	in left inlet: deactivate the current outlet and activate the outlet specified by the integer parameter. If the parameter is zero (or negative) no outlet will be activated. If the parameter is larger than the number of outlets, the rightmost outlet will be activated. in right inlet: simply pass message through current outlet (or all outlets if in a Spray Mode) without interpretation.
spray	in left inlet: Spray, followed by any parameters, will cause ice.pump to turn the parameters into a message that will be sent through all



outlets in right-to-left order. This essentially sets ice.pump into Spray Anything Mode for a single message (see Spray Modes, above/below/under/over/sideways/down/backwards/forward/square/round/dowee-ouwa-ou-wa-ou-ou-wa)  
in right inlet: simply pass message through current outlet (or all outlets if in a Spray Mode) without interpretation.

anything

anything

tattle

Double-clicking on a ice.pump object box, or sending the tattle dbclick message to the left inlet, will cause internal state information about the ice.pump object to be printed in the Max window. [note to self: for production version, limit info to stuff the user might possibly, conceivably ever want to know)

checkconnections

<<hell, another internal message... using internal messages is the only way to get some things done, but nobody wants to know and nobody wants to document it anyway.>>

DSPaudio, Inc.

Thank you for using iCE! We encourage our users to contact us and share experiences, insights and requests with our team in an effort to further the art of computer music.

[signal@dspaudio.com](mailto:signal@dspaudio.com)

3650 18<sup>th</sup> St. Suite 1  
San Francisco, CA  
94110

<http://www.dspaudio.com>

