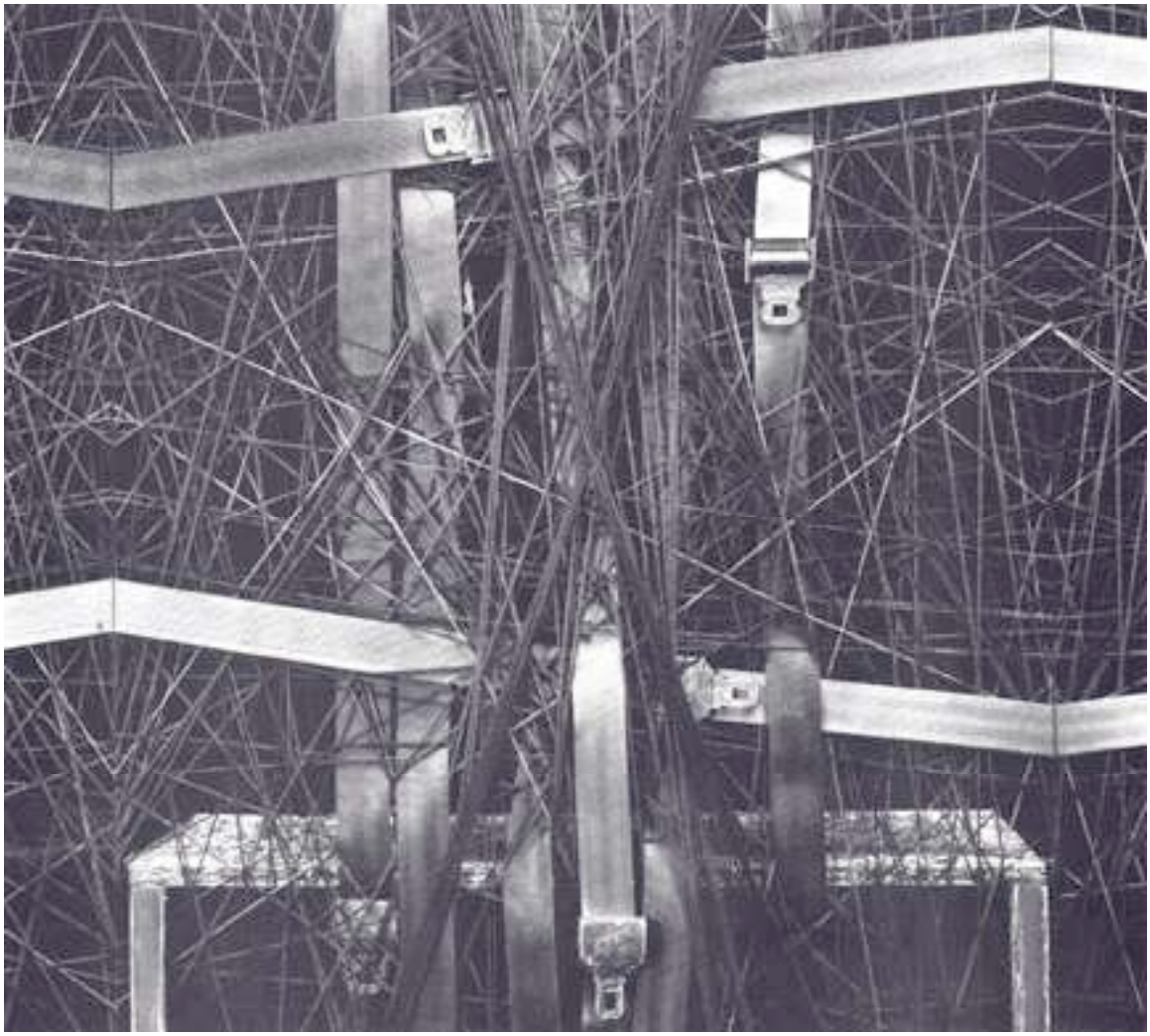
MAX



Getting Started

Copyright and Trademark Notices

Credits

Content: David Zicarelli, Gregory Taylor

Cover Design: Lilli Wessling Hart

Graphic Design: Gregory Taylor

Table of Contents

Table of Contents

Table of Contents

Introduction

What is Max?

> "Max is a graphical music programming environment for people who have hit the limits of the usual sequencer and voicing programs for MIDI equipment."
>
> —Miller Puckette, Max reference manual, 1988

Max was conceived in 1986 as a project for producing interactive music at IRCAM (Institut de Recherche et de Coördination Acoustique/Musique) in Paris. The original author was Miller Puckette. Max became a commercial product from Opcode Systems in 1991 with further development by Puckette and David Zicarelli. Cycling '74 became the publisher of Max in 2000. Since that time, Max expanded to include audio data (with the introduction of MSP) and image/matrix data (with the introduction of Jitter).

Max lets you control your equipment in any way you want. You can create applications for composing, improvising, and ordering or modifying media—anything you can imagine doing with a computer. Because Max turns all control information into a simple stream of numbers, you can "patch" anything to anything else.

Max provides you with a high level, graphical programming language. Programs are "written" using graphical objects rather than text. This reduces the need to learn a lot of arcane commands and syntax, and it provides a clear and intuitive way to write programs simply by connecting objects to each other.

Max takes care of all the low level programming tasks for you. It will trigger events at any arbitrary time in the future, interface to MIDI and other communication protocols, and perform useful logical operations.

Applications made with Max run in real time. Because of its speed, Max enables you to write programs that generate music instantly based on what you play, or that modify your performance as you play.

Max is based on the C programming language. Max provides a simple yet versatile, high level, graphical language which is itself written in C, but will be easy to use for those familiar with almost any other programming language, or even for those who have never programmed before. For those who are fluent in the C language, however, Max can be combined with C code that you write. So, if there's something you need to do that Max can't do—and Max can do a lot—you can write your own Max objects in C. The release of Max 4.5 further extends Max's capabilities by supporting Java and Javascript.

Introduction

How To Use The Manuals

The Max User's Manual consists of three volumes: **Getting Started**, **Tutorials and Topics**, and the **Max Reference Manual**.

This volume, **Getting Started**, includes:

Setup — Read this section first.

Tells you what equipment and System software you will need to use Max successfully, how to connect your MIDI devices, how to install the Max application and how to get technical support.

Overview — Read this section next if you have never used Max before.

Contains a description of the Max application, introduces basic terms that will be used throughout the manuals, and tells you how to use online help.

Elements of Max

Contains information on menu commands, keyboard shortcuts, and Max objects.

Max and MIDI

Contains information on the MIDI software protocol, receiving and transmitting MIDI, using Max with CoreMIDI (Macintosh) or MME (Windows).

The second volume, **Tutorials and Topics**, provides a step-by-step course on how to program with Max and a collection of discussions of certain topics unique to programming with Max.This volume includes:

Max Tutorial — Read this section next if you have never used Max before.

Takes you through a step-by-step course on how to program with Max. Each chapter in the Tutorial is accompanied by an instructive program written in Max, found in the Max Tutorial folder.

Max Topics

Contains discussions on issues of programming—data structures, loops, encapsulation, debugging, graphics, making standalone applications, etc.—and explains specifically how those issues are handled in Max.

Introduction

The third volume, **Max Reference Manual**, should be referred to as needed for detailed information on specific Max objects.

Max Objects

Contains precise technical information on the workings of each of the built-in and external objects supplied with Max, organized in alphabetical order.

Max Object Thesaurus

Consists of a reverse index of Max objects, alphabetized by keyword rather than by object name. Use this Thesaurus when you want to know what object(s) are appropriate for the task you are trying to accomplish, then look up those objects by name in the *Objects* section.

The document **Javascript in Max** describes support for the Javascript language, and is intended to supplement the tutorials found in the Tutorials and Topics manual.

Your Max distribution also includes a set of tutorials and manuals for using Java for Max development.

If you are interested in C-language programming, the additional document **Writing External Objects for Max**, explains how to create your own external objects for Max using the C programming language. The document can be downloaded from http://www.cycling74.com.

Most users of Max will not need to read the **Writing External Objects for Max** document. However, if you find you want to add additional functionality to Max, and you know how to program in C, the document contains complete instructions on how to do so, and sample source code is available in the *Max SDK* folder. The instructions assume a thorough working knowledge of the C programming language.

If you are new to Max, we suggest you begin by reading the *Setup and Overview* sections of this manual, then trying a few of the Tutorials. You can also learn by looking at the help files in the *max-help* folder located at C:\Program Files\Cycling '74\MaxMSP 4.5\max-help on Windows or /Applications/Max4.5/max-help on Macintosh and, and by browsing the Max Object Thesaurus and some of the expository chapters in the **Tutorials and Topics** Manual.

Introduction

Manual Conventions

The central building block of Max is the object. Names of objects are always displayed in bold type, **like this**.

Messages (pieces of information that are passed to and from objects) are displayed in plain type, like this.

In the "See Also" sections, anything in regular type is a reference to a section of this manual, the **Tutorials and Topics** manual, or the **Max Reference Manual**.

Other Resources for Max Users

The help files found in the *max- help* folder provide interactive examples of the use of each Max object.

The Cycling '74 web site (*http://www.cycling74.com)* provides the latest updates to our software as well as an extensive list of frequently asked questions and other support information.

Cycling '74 runs an internet mailing list where you can ask questions about programming, exchange ideas, and find out about new objects and examples other users are sharing. For information on joining the discussion, as well as a guide to third-party Max-related resources, visit *http://www.cycling74.com/community/*

Finally, if you're having trouble with the operation of Max, send e-mail to *support@cycling74.com*, and we'll try to help you. We'd like to encourage you to submit questions of a more conceptual nature ("how do I...?") to the mailing list, so that the entire community can provide input and benefit from the discussion.

System Requirements, Installation, and Authorization

See the file Read Me Before Installing for the minimum system requirements, installation instructions, and authorization procedure.
Connecting MIDI Equipment

> Note: This section covers the use of external MIDI gear with your computer. If you aren't planning to use MIDI communication with Max, you don't need to read this section.

1. Make sure your MIDI interface is connected to the appropriate serial or USB port on your computer.

2. Connect the MIDI Out of the MIDI controller(s) to the MIDI In of the interface to send incoming MIDI messages to Max.

3. Connect the MIDI Out of the interface to the MIDI In of the sound source(s) to send output MIDI messages from Max.

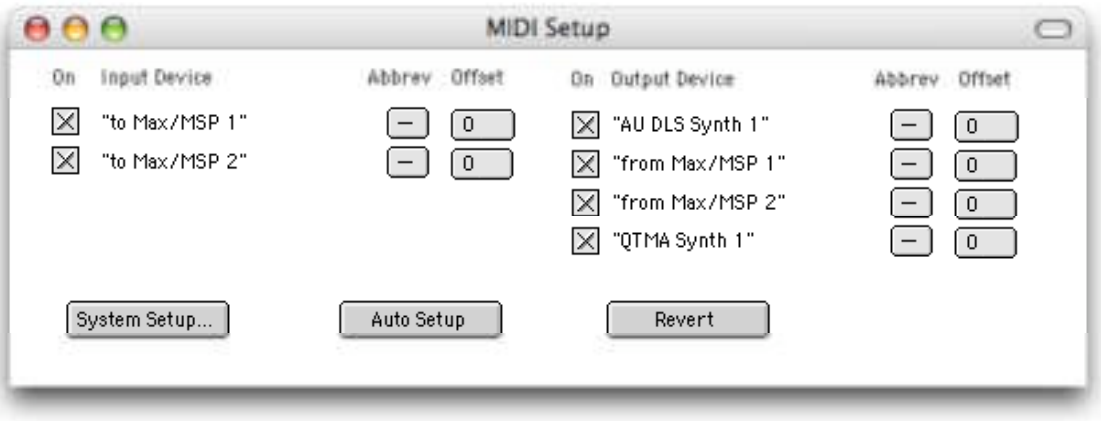
4. The Macintosh OS comes equipped with a built-in MIDI system called CoreMIDI. You can use the *Audio MIDI Setup* application found in /Applications/Utilities to specify the ports, interfaces and MIDI devices in your MIDI setup.

On Windows, all MIDI devices which are installed correctly on your system and appear in the Sounds and Audio Devices Properties (Start - Settings - Sounds and Audio Devices) are available to Max/MSP for MIDI I/O.

Setup

MIDI Setup Dialog

If you choose **Midi Setup** from the File menu), you will be presented with a window showing you the Input and Output devices available:



Max's MIDI Setup dialog box is used to assign abbreviations and MIDI Channel Offsets to each of your Input and Output Devices in your MIDI Setup. The *on/off* toggle is used to enable or disable a MIDI input or output. The *abbreviation* is for your convenience. You may wish to specify a specific device in creating a Max MIDI object, and it's much faster to type the letter b than *Kurzweil 1000PX*. The Channel Offset is used to distinguish MIDI devices by their MIDI channel. For example, a Channel Offset of 16 sets an MIDI Output Device to receive MIDI on Channel 1 when you specify channel 17 in Max. Setting the Channel Offset for a MIDI Input Device lets you determine the source of a MIDI message. For example, if the message is marked as being on MIDI Channel 17, you know it comes from MIDI channel 1 of a device with a Channel Offset of 16.

It is not essential that you configure this information before using Max for the first time, but MIDI input and output may not be predictable unless you do so. The Using MIDI chapter of this manual provide complete information about using this dialog box.

The default MIDI output device under Max is the internal synthesizer supported by your operating system. On the Macintosh, this is an AudioUnit DLS synthesizer that supports both its own set of internal sounds (as a General MIDI bank), as well as Level 2 SoundFont files.On Windows, this is the Microsoft DirectMusic DLS synthesizer. Note that the Microsoft DLS synthesizer does not support SoundFont files. For more information about working with DLS synths, see the section "Using the DLS Synthesizer" located in the chapter **Using Max with MIDI.**

Setup

For now, you may wish to click *Auto Setup*, which will assign unique abbreviations and MIDI Channel Offsets to all devices in your system that are connected to different MIDI cables. Devices at the top of the list will be assigned an abbreviation and a Channel Offset of 0. You should set the Channel Offset of your most commonly used Input and Output Device to 0. This will let you address it with the most common range of MIDI Channels, 1-16. If Auto Setup did not do this, you may wish to change the assignments.

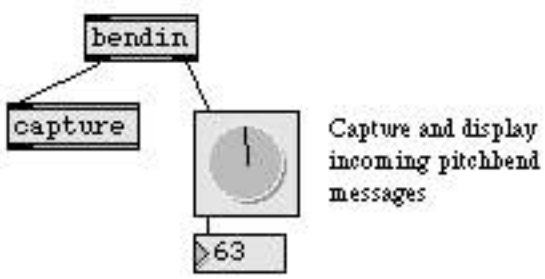
Introduction

This chapter briefly describes the Max application and introduces basic terms that will be used throughout the manuals.

If you have relatively little experience with computer programming or MIDI, you might not immediately understand everything you read in this chapter. Never fear. The Tutorial chapters in the Max **Tutorials and Topics** manual will cover topics in much greater detail, starting from square one. Max is a complex application and we don't expect everyone to "get" everything immediately. This overview will simply familiarize you with the application and its basic elements.

Programming With Objects

When you work with Max, you'll be developing applications *graphically* using objects, which will appear on your screen as boxes that contain either text or icons. You connect objects together to create a working program.

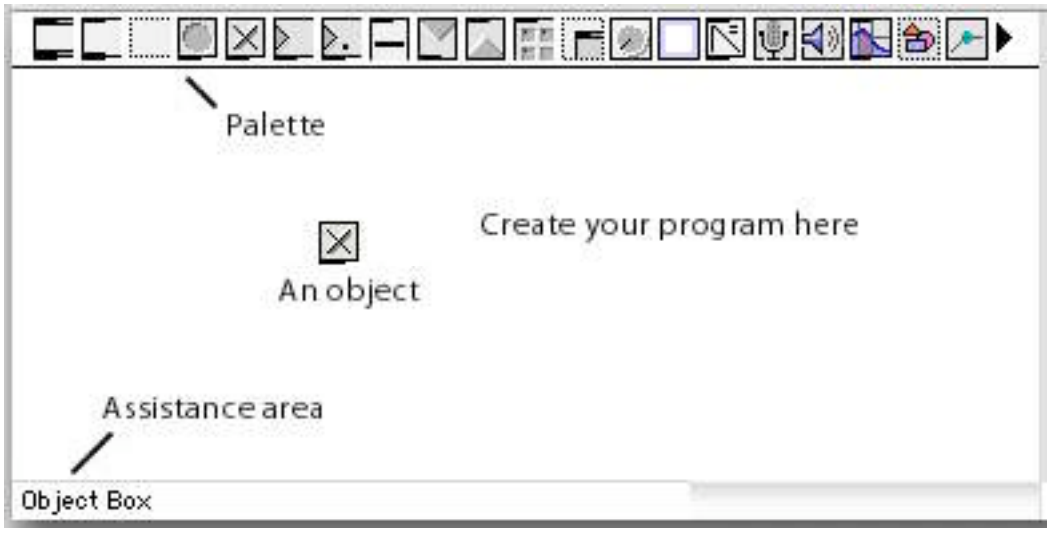


A simple program written graphically in Max

Max has about 200 different objects (and MSP adds about 200 more for audio processing, and Jitter adds an addition 140 objects for matrix and video/image processing), each of which performs one or more specific tasks. In addition, you can write your own program in Max, save it, and then use that program as an object inside other programs that you write.

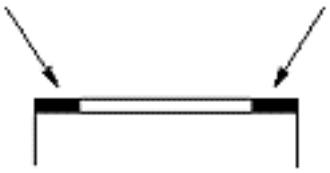
Overview

You create a Max program in a *Patcher* window, which works much like a drawing program. You select an object from a palette, then click where you want the object to go.



Objects are different types of boxes, like this:

Objects have *inlets* at the top, used to receive information from other objects...



…and *outlets* at the bottom, used to send information to other objects.

You connect two objects together with a *patch cord*. Always drag from the outlet of one object to the inlet of the other. When dragging a patch cord, you can be anywhere inside the object box and the inlet will "expand" to indicate that you can release the mouse button to make the connection.

Inlet expanding to accept a patch cord

Normal Objects and User Interface Objects

The most common type of object is the *object box*, which has two lines at its top and bottom.

The action performed by an object box depends on the name you type into it. The name is like a verb, describing what the object will do. The name will either be a single word, such as **makenote**, or a symbol such as **+** (add) or **>** (is greater than).

Typically, objects send information out their outlets in response to information they receive in their inlets. Each inlet and outlet of an object has a different purpose. These are described in detail in the *Objects* section of the **Max Reference Manual**. A brief description of the use of each inlet and outlet is available in the *Assistance* area of the Patcher window, as shown in the next example. Just move the mouse over the inlet, and descriptive text appears in the Assistance area.

Getting Assistance for the right inlet of the **delay** *object*

A few objects lack either inlets or outlets because they receive or send information from some- where outside of the patch that contains them. For example, the **midiout** object has no outlets because it transmits values directly to MIDI.

Other objects are *user interface* objects. They look like buttons, dials, sliders, keyboards, etc., and respond not only to messages received from other objects, but also to clicking and dragging with the mouse. They let you control the program by using the mouse, and also provide a means of displaying numbers and other messages onscreen in a variety of ways. The name of each object is displayed in the assistance area when the mouse rolls over them.

There are some objects, such as **comment**, that do nothing. The comment object lets you put notes to yourself reminding you what you have done, or notes to others telling them how the program works. You can also use comment objects to label user interface items to tell other people how to use your program.

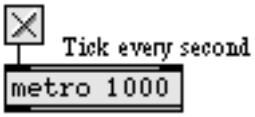
When you've finished making your program, you can save it as a Max document, otherwise known as a *patch*. If you give your patch a single-word filename (that doesn't begin with a number), you can then use it as an object within another Max patch just by typing the filename into an object box.

For a more detailed explanation of Objects and related issues, you can refer to the chapters titled *Objects* and Externals in this manual and the chapter titled *Encapsulation* in the **Tutorials and Topics** manual.

Arguments

Often additional words or numbers are included in an object box after the object name to provide initial information to the object or to specify some of its characteristics. The additional words are known as *arguments* to the object.

When you create a metronome object **metro**, for example, you may type in a number argument after the message name to specify how many milliseconds the metronome should wait between ticks.



A few objects have *obligatory* arguments, while others have *optional* arguments. If you forget to type an obligatory argument into an object box, Max will print an error message in the

Max window advising you of that fact, and will refuse to create the object. If you choose not to type in an optional argument, Max provides a default value for that argument. The type of arguments that can be given to each object—and their default values—are detailed in the *Objects* section of the **Max Reference Manual**.

Messages

What actually gets sent through the patch cords? A *message* is the information passed from one object to another. A message can be a number, a *list* of numbers separated by spaces, a word (referred to in Max as a *symbol*), or any arbitrary combination of words and numbers. The contents of a message determine its *type*.

The types of Max messages are:

int — When a message consists of nothing but an integer number (such as the message 127), it is understood by Max to be a message of type int. A message could also say int 127.

float — When a message consists of nothing but a number containing a decimal point (such as the messages 3.97 or 3. or .97), it is understood to be a message of type float (short for "floating-point number"). The decimal point lets Max know it is a float. As with ints, it is possible—but not necessary—for a message to say float 3.97.

list — A list of two or more numbers separated by spaces (such as 60 79 1.02 4) is of the type list. The message need not (but may) begin with the word list. Max recognizes a list whenever it sees a message consisting of a number followed by anything. In fact, a list can contain words as well as numbers (as in the message 1 start 768), so long as it *begins* with a number. A list can contain up to 256 items.

bang — The message bang is a special message meaning "do whatever it is you do." For example, when the **random** object receives the message bang, it sends a randomly chosen number out its outlet.

symbol — A symbol is a word or other non-numeric collection of characters. Many symbols are meaningful commands when received by certain objects. The exact symbols understood by each object are listed in the *Objects* section of the **Max Reference Manual**. For example, the **seq** object, a sequencer for recording MIDI performances, responds to messages received in its inlet such as start, stop, record, delay, and print. All of those messages would be meaningless to an object like * (multiply), which expects only numbers (or bang) in its inlets.

any message — A message can, in fact, consist of *any* combination of words and numbers. Some objects can handle any type of message. Most objects, however, expect to receive one of the above types of message, and will not understand other messages.

When an object receives a message it doesn't understand, it will either ignore the message entirely or will print an error report. In many cases, Max even stops you from making such mistakes while you are editing a program. As you connect an outlet of one object to the inlet of another object, Max does its best to analyze what message will be traveling through the patch cord. Max won't let you connect an outlet to an inlet that won't understand the message. If an inlet is not highlighted when you try to connect a patch cord to it, Max will not make the connection.

Attributes

In version 4.5 and later, a few Max/MSP objects—notably, **mxj** and the pattr family of Max objects **pattr**, **pattrstorage**, **pattrhub**, and **autopattr**)—use *attributes*. Attributes are another way to specify the behavior of Max objects, and are found and used widely in Jitter—an extension to Max and MSP that allows you to work with matrix-based data, including video and OpenGL.

If you already own or use Jitter, you will be familiar with how they work. This section will provide an explanation for users who have not encountered attributes.

You can use attributes to initialize, change, and find out the current values stored in an object, and the attachment of each value to a fixed attribute name means that you don't need to remember the ordering typed-in arguments or what each inlet in a complex object does. Future versions of Max/ MSP will implement attributes for Max/MSP objects much more widely.

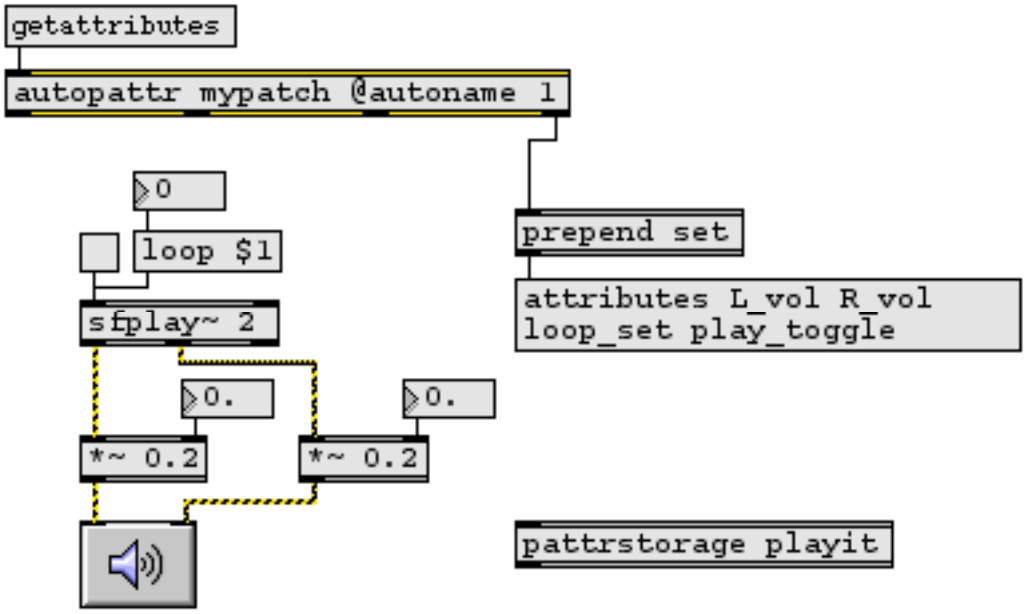
As with arguments, you can type attributes into an object box along with the object's name, or you can set (and retrieve) attributes using Max messages after the object is created. Typed-in attributes are set in object boxes by using the @ symbol followed by the name of the attribute and one or more arguments (which could be any kind of Max data: ints, floats, symbols, or lists). You can enter as many attributes as the object recognizes, in any order, after the object's name.

You can type in attributes, or set them using Max messages

Note that there is no space between the @ sign and the name of the typed-in attribute you want to set. The @ character tells the object to interpret the word attached to it as an attribute name instead of an argument value for a previous attribute. Objects can have both typed-in attributes and typed-in arguments—but the arguments must be listed *first*.

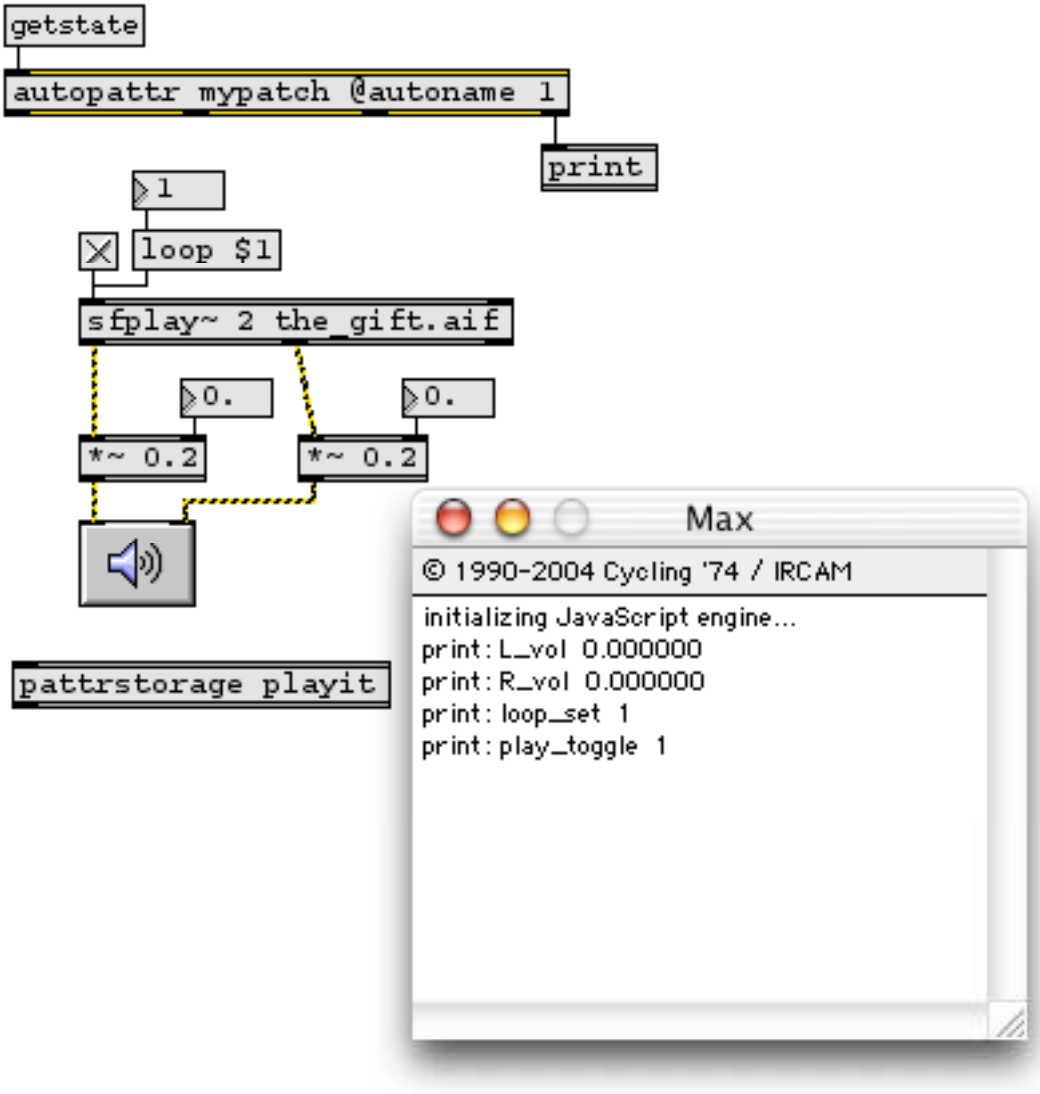
An additional (and very useful) feature of attributes is that you can ask an object with attributes to tell you about the attributes it knows about, and what values it currently has stored for any given attribute.

The getattributes message causes any object with attributes to output the message attributes followed by a list of all the attribute symbols that the object understands.



What attributes does an object have?

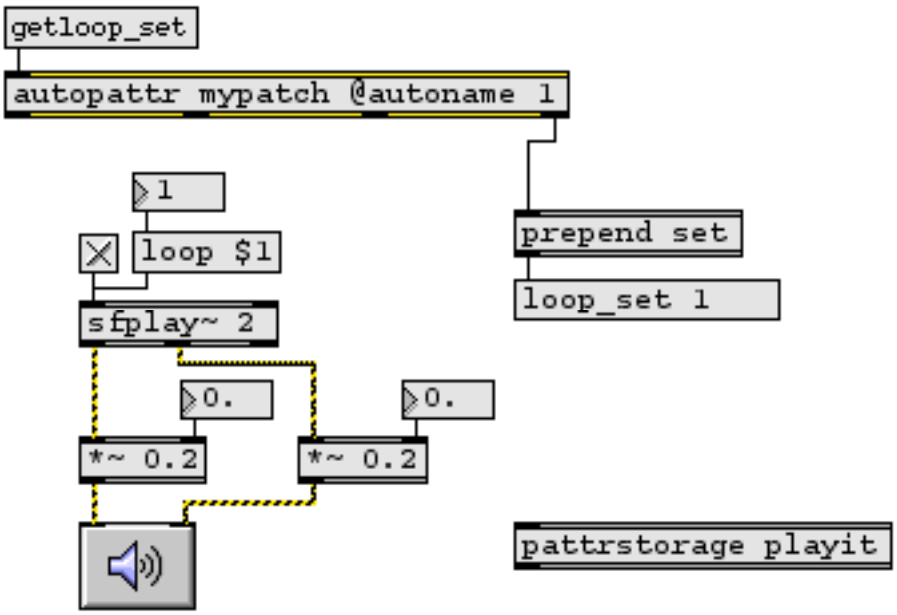
The getstate message dumps out all the attributes for the object as if every conceivable attribute query had been performed all at once.



Finding an object's state

You can then use **route**, **unpack**, and other Max objects to extract the attributes as you need them.

You can also check on the value of any individual attribute using the Max message get followed (with no space) by the name of the attribute you want to know about. The resulting value is output by the object as a message (beginning with the attribute's name), sent out the object's right outlet. Using this message, you can discover the current value of an attribute, even if you never set the attribute in the first place.



For more information on attributes and the way they are used in specific objects, see the Max Reference Manual pages for the **pattr**, **pattrstorage**, **pattrhub, autopattr**, and **patcherargs** objects.

Numbers

Max distinguishes integer numbers from decimal numbers (with a fractional part). Integer numbers in Max are a data type called *int*, and decimal numbers are a data type called *float*.

When a number is converted from float to int by Max, the fractional part of the float is chopped off (*truncated*) rather than rounded up. Thus, when the number 4.1 is converted to an int, it becomes 4. Likewise, the number 4.999999 becomes 4 when converted to an int.

When you write programs that deal with MIDI information, you'll generally use ints. When dealing with audio information, you'll probably use more floats. MIDI data is always expressed in terms of ints in Max. Time is usually expressed in terms of milliseconds in Max, which can be either int or float.

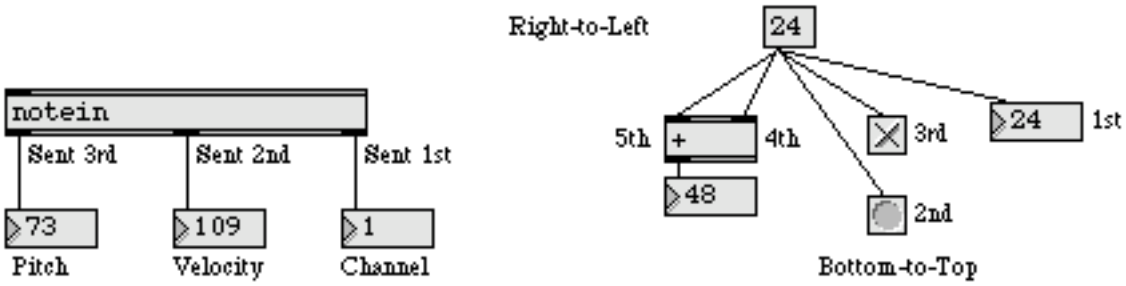
You can type in any int as a hexadecimal number. Hexadecimal is a base-16 number system

often used in MIDI specifications and other computer-related documentation. Precede a hexadecimal number with 0x (zero plus a small x), as in 0xF0 (same as the decimal number 240).

Message Order

An object often has more than one outlet, and usually sends messages out all outlets "at the same time." Actually, nothing *really* happens at the same time in Max. Things can happen so fast as to seem simultaneous, but it's important to know how Max orders messages.

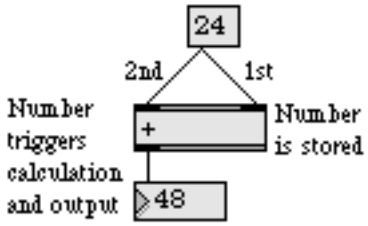
When an object sends several messages "at once", out different outlets, the order is actually *right- to-left*. The rightmost outlet sends its message out first, then the outlet just to the left of that, and so on until the leftmost outlet. This is true of virtually *every* object in Max.

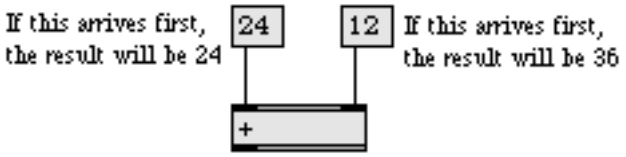


The example on the left illustrates the right-to-left order of message output. In the example on the right, a single outlet has many patch cords connected to it, all leading to different inlets of other objects. In this case, the order that messages are sent is determined by the *right-to-left* screen position of the receiving objects. If two receiving objects are perfectly aligned vertically (neither is further right than the other), the order is *bottom-to-top*; the object that is lower on the screen receives the message first. If a single outlet is connected to two inlets of the same receiving object, the rightmost inlet will receive the message first.

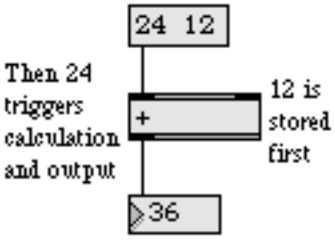
When an object has more than one inlet, it *expects* to receive messages in its inlets in right-to-left order. Generally, an object will store all the messages it receives *until* it receives a message in its leftmost inlet, then it will perform some operation and send messages out its outlets. The vast majority of objects are "triggered" by a message in their leftmost inlet. There are some exceptions to this rule, but the left inlet is the triggering inlet for almost all objects.

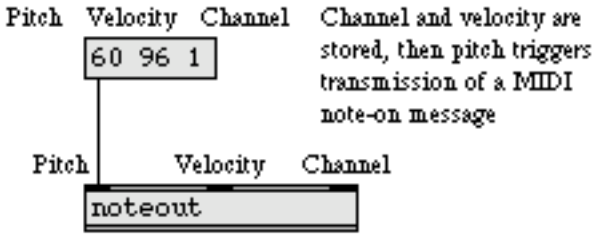
Notice that this makes sense because objects send their messages right-to-left, so the last message an object receives will usually be the one that triggers output.





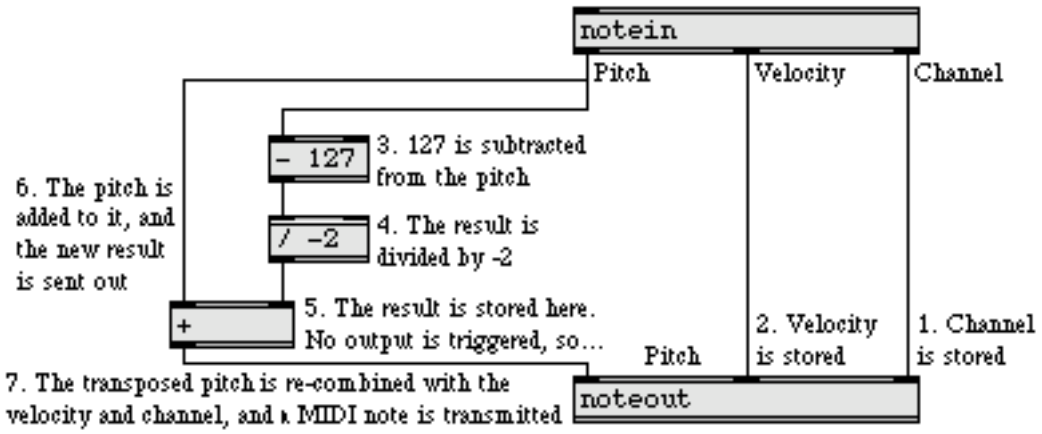
Some objects can receive a list of numbers in the left inlet, and the result will be the same as if the numbers had been received in different inlets in right-to-left order.





Finally, suppose that a message is sent several places from the same outlet, but that message triggers the receiving objects to send *their* messages. What is the order in that case? The answer is that the message will trigger the receiving object, and that object's output will be sent next. The messages will continue triggering other objects until a message is sent that triggers no other action, then Max goes back to the original sending object and sends its next message.

In the example below, note that steps 3, 4, and 5 are all performed (up to the point where no further message is triggered) before the message is sent to the left inlet of the **+** object.

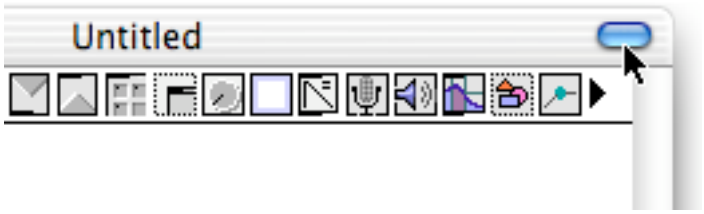


Windows

Max has six main window types: Patcher, Text, Table, Timeline, the Max window, and the New Object List. You can open as many of the first four types of window as you like, but there is only one New Object List and one Max window. Other objects such as env and movie open their own windows.

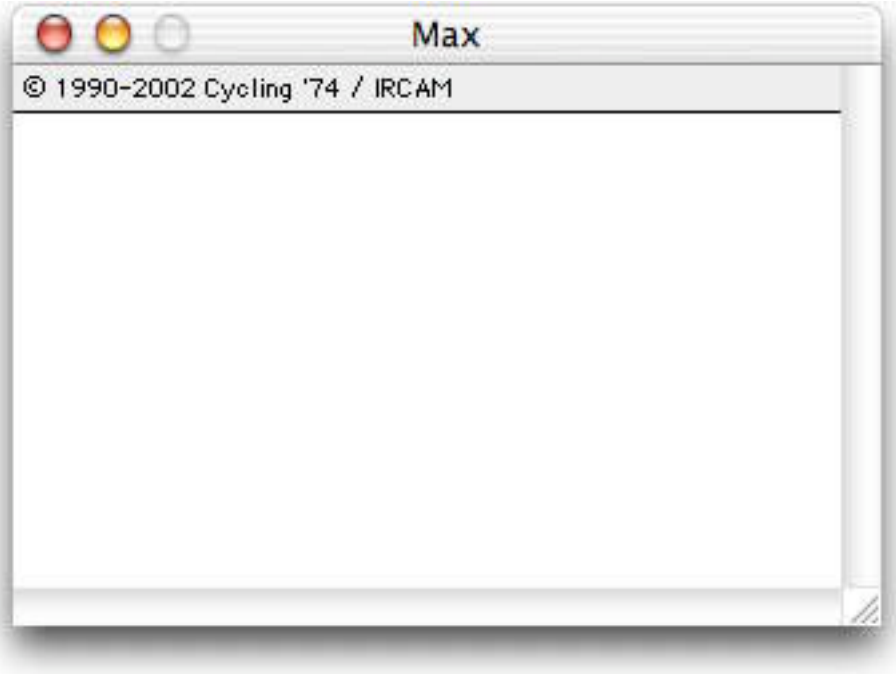
On Windows, each Max window contain the standard icons for minimizing, maximizing and closing the window. Macintosh windows contain the standard icons for closing a window, sending it to the dock, and minimizing/maximizing the window.

Macintosh versions also include a transparent rectangular pill on the right side of the window that can be used to toggle between locked and unlocked Patcher states.



Max Window

When you open the Max application, you'll see the Max window first. This is the place Max prints out messages to the user.



After you launch Max, you may see a few messages from some external objects that were loaded during the startup process.

Any error messages or warnings generated while you are editing or running a program will appear in the Max window. As you're debugging the programs you write, you can have Max print out exactly what is flowing through the patch cords at any given moment, to see if your patch is working properly. For more information on printing in the Max window, see the *Debugging* chapter in the **Tutorials and Topics** Manual.

Patcher Window

Once the Max application has loaded, you begin programming by creating a new Patcher window (by choosing Patcher from the **New** submenu of the File menu) or by opening an already existing patch (with the **Open…** command in the File menu).

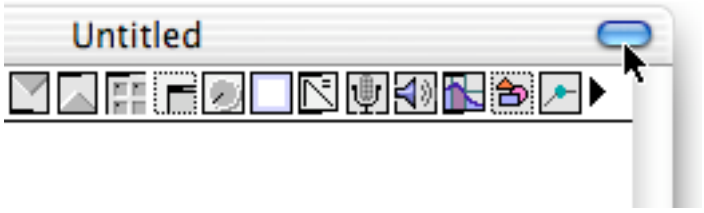
Programs are "written" in Max by placing objects in a Patcher window and connecting them with patch cords to make patches. Once you have created the patch, you can run it and ensure that it actually does what you want it to do. You can save the patch as a Max document, to run or edit again later.

A patcher window is either locked or unlocked. When it is unlocked, you can edit the patcher by moving objects around, creating new ones, and connecting objects together. When it is locked, you are operating the patcher by clicking on objects (such as sliders) that do things.

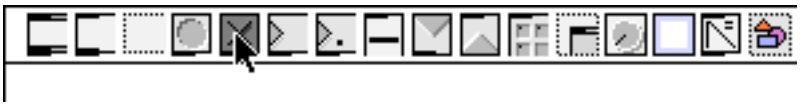
The lock/unlock state of the window is indicated by the presence of the patcher palette at the top of the window. If you see the palette, then the window is unlocked.

There are several other ways you can lock or unlock a patcher.

- Choose **Edit** from the View menu, or type Command-E on Macintosh or Control-E on Windows.
- Command-click on Macintosh or Control-click on Windows on the "white space" in the Patcher window.
- On Macintosh, there is a transparent rectangular pill on the right side of the window that can be used for toggling between locked and unlocked state.



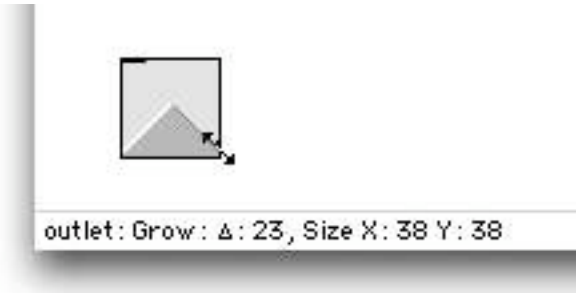
When you create a new Patcher window, it will already be unlocked. To place an object in the window, click on the desired icon in the palette.



The cursor will become the same as the palette icon.

Then click in the Patcher window where you want to place the object.

You can move one or more objects to a new location by selecting and dragging them with the mouse. You can select objects for editing operations such as **Cut**, **Copy**, and **Duplicate**, or if an object box contains text you can change its font or font size by using the Font menu. You can also resize an object by clicking on its grow bar (the tiny rectangle in the lower-right corner of the object). The cursor will change to a two-way or four-way arrow when you are above the grow bar (which is sometimes invisible). As you resize the object, the assistance window will show you the change in the size of the object, and its current dimensions, in pixels.

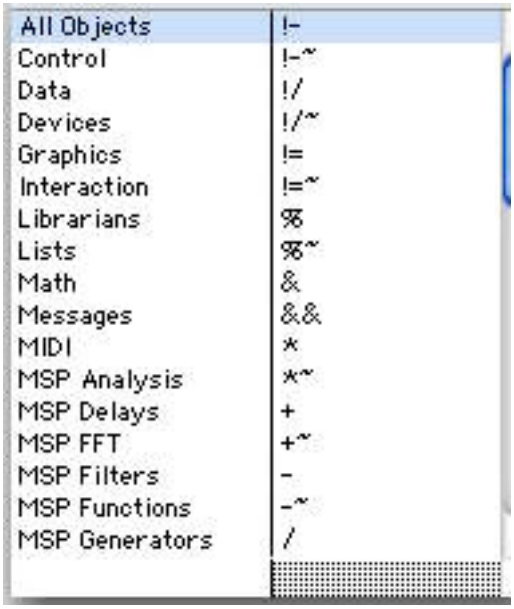


At any time you can lock the Patcher window and use your patch. When the window is locked, the object palette disappears, and clicking on certain objects now operates them as parts of the user interface. When you have finished editing your patch, and have tested it to

see that it runs properly, save it by choosing **Save As...** from the File menu. The next time you open that Max document, it will open locked and ready to use.

New Object List

When you add an object box (the leftmost icon in the palette) to a Patcher, Max opens the New Object List window (if **New Object List** is checked in the Options menu). The window is divided into two columns. The left column lists categories of objects, and the right column lists the names of objects within the selected category.



You can scroll through the list of objects and select one with the mouse, or you can use the computer keyboard to type in the first few letters of an object name as you would in a standard "open file" dialog. (This can be a useful shortcut for typing the entire name of an object, especially when the current category is All Objects.)

When the object name you want is selected, double-click on it, or press Return, Enter, or Space, and the name will be entered into the object box. If you want to open a help file for an object, Option-double-click on its name in the list on Macintosh, or Alt-double-click on Windows. If you don't want any of the names shown in the New Object List, press the Delete (Backspace) key, and the window will go away without entering any text into your box.

Text Window

In addition to Patcher windows, you can open simple Text windows for taking notes or for editing the data inside certain objects. Create a new Text window by choosing **Text** from the **New** submenu of the File menu.

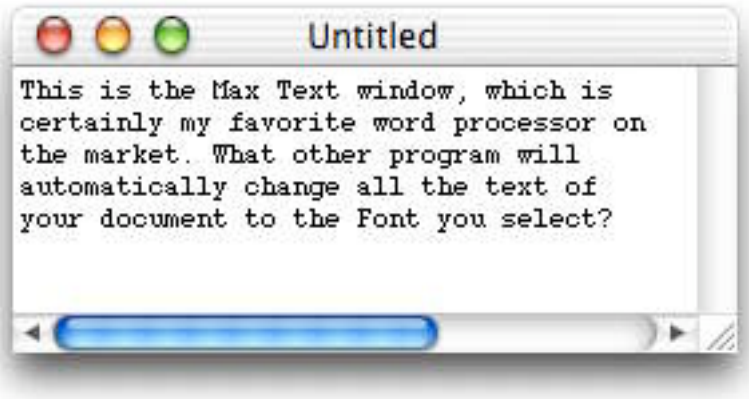
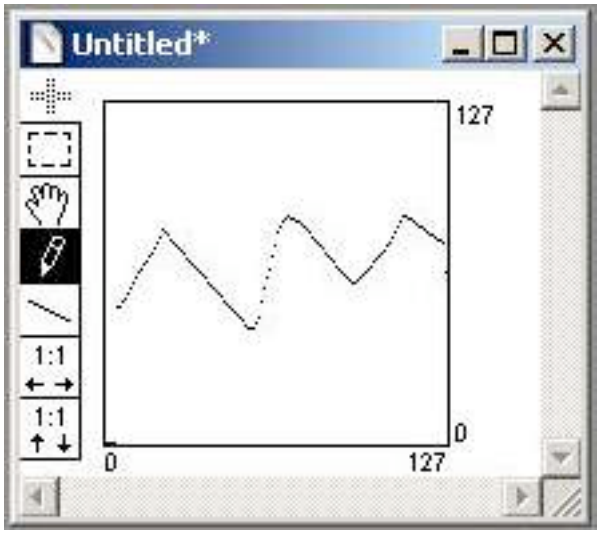


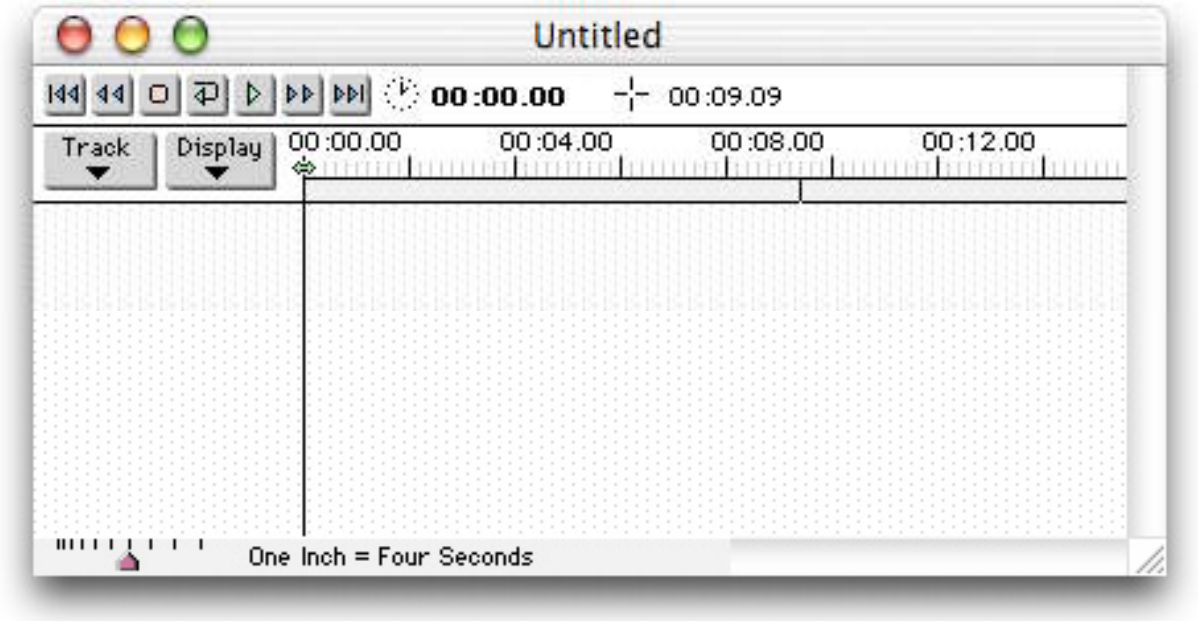
Table Window

The Table window is used to view and edit an array of numbers. The numbers displayed in such a window may be contained in a corresponding **table** object that you will place in a Patcher window to access the data.



Timeline Window

A Timeline is a graphic arrangement of Max messages, to be sent to specific objects in specific patches at specific times. The timeline can be played using the controls in the Timeline window, or it can be played from within a patch that contains a **timeline** object. Create a new Timeline window by choosing **Timeline** from the **New** submenu of the File menu, or by typing timeline into an object box.



For more information, see the *Timeline* chapter in the **Tutorials and Topics** Manual.

Max Documents

Whenever Max tries to find a document automatically, it looks first in the folder that contains the patch that's being loaded, then it looks in the folders specified in the File Preferences window (accessed by choosing **File Preferences**… from the Options menu.), then it looks in the folder containing the Max application. This searching order is known as the *file search path*.

There are many kinds of Max documents. The Max application is accompanied by:

- The *max-startup* folder containing external objects (objects that are not contained within the application itself) which Max loads into memory automatically when it starts up.
- The *externals* folder containing additional external objects.

- The *max-help* folder containing example patches demonstrating how each object works.
- The *Max Tutorial* folder containing example patches to accompany the *Max Tutorial* sections of the **Tutorials and Topics** Manual. (These patches are not essential to the Max application.)
- The *patches* folder containing folders which hold Max editors and preference files (*editors*), the Inspectors used to set the attributes for some Max and MSP objects (*inspectors*), graphic images used for some UI objects (*picts*), and useful utility and guide patches (*extras*).
- The *tiAction* folder for containing timeline "action" patches.

Note: If you're using Max/MSP, there are some additional folders created for audio-specific purposes. Refer to the MSP documentation for information on these folders.

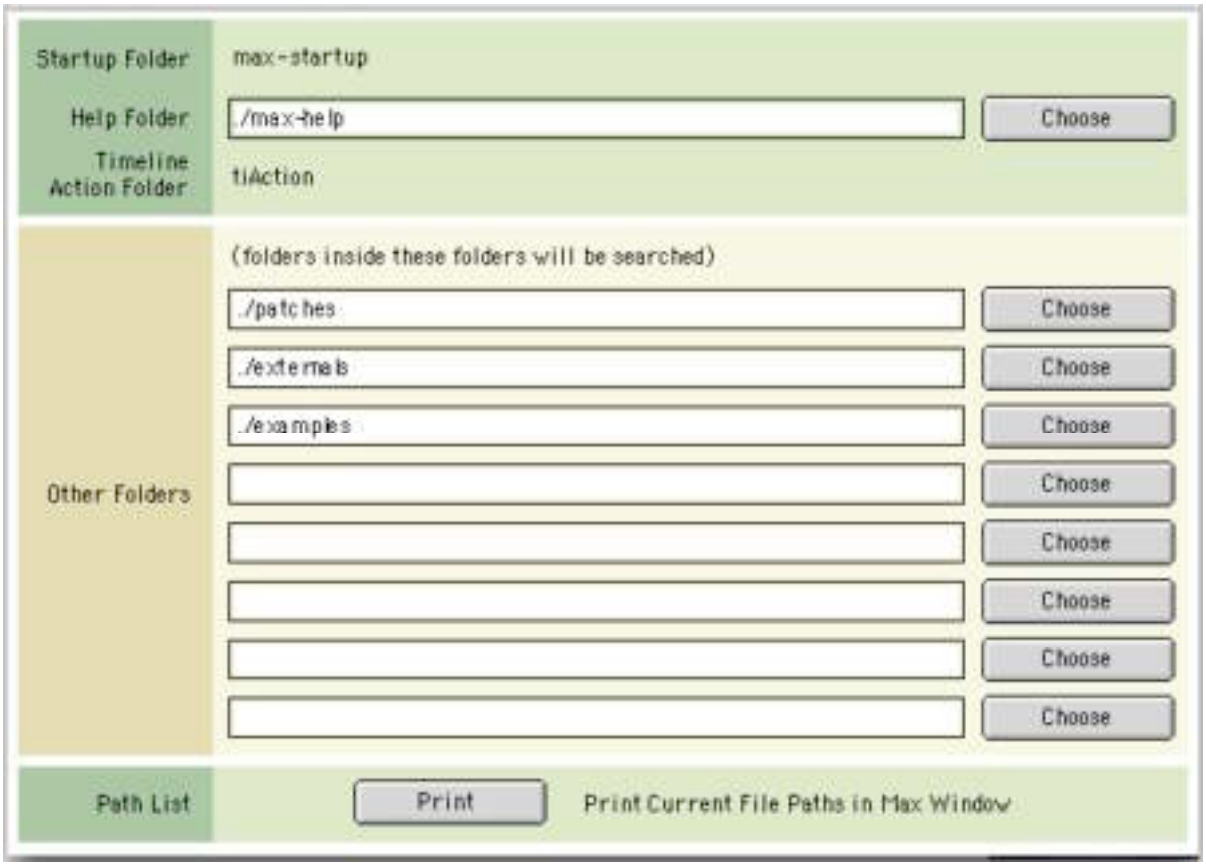
We hope you'll be using Max to create a variety of documents yourself:

- Patches you construct in the Patcher window. These are the documents you will create most often; they are working programs you have written in Max.
- Binary or text files containing data for such objects as **coll**, **funbuff**, **mtr**, **preset**, and **table**.
- MIDI files saved by the **seq** or **detonate** objects.
- Script files describing envelope functions.
- Timelines and timeline action patches.

In order to keep track of all these documents, Max lets you specify the names of folders that it should use when it looks for various files. To name these folders, choose **File Preferences**… from the Options menu.

The **File Preferences** window displays the name of the important folders used by the program. The startup folder, from which external objects are automatically loaded when the application is opened is called *max-startup*. The folder for help files is called *max-help*. The timeline action folder, where files used in timelines are stored is called *tiAction*. You can specify six other folders (and their subfolders) for Max to examine when loading a file.

This is the File Preferences window:



If a folder name begins with a dot-slash, it is a subfolder of the folder that contains the Max application. Otherwise, the name must start with the name of the volume (disk drive) which contains the folder.

By default, Max will look for patches, external objects, tables, and other files in folders called ./externals, ./patches and ./examples. You can specify additional folders by adding their names in slots below these names.

Max Preferences

Max remembers many of your working preferences by storing settings each time you quit the application. The first time you use Max, a folder containing your preferences is created.

On Macintosh, this folder is called “Max 4 Preferences Folder”, and is in the Preferences folder inside the Library folder of your home directory.

On Windows, this folder is called “Max 4 Preferences Files”, and its location varies according

to the name of the user logged in (e.g. C:\Documents and Settings\<user>\Application Data\Cycling '74\Max 4 Preferences Files\).

This is where Max stores settings you have chosen in the **Midi Setup** and **File Preferences** windows, as well as the current settings of the Options menu.

In addition, you can set a default font and font size for the Max window and subsequent new Patcher windows by using the Font menu while the Max window is the active window. Those font characteristics will be stored in the preferences file.

MIDI

Max has many built-in objects for transmitting and receiving MIDI data to and from the ports available in your MIDI system. Objects that transmit MIDI messages have no outlets, since they send MIDI data out from the Max application. Similarly, objects that receive MIDI messages from the outside world get their input from MIDI devices, rather than from other Max objects.

The **midiin** and **midiout** objects receive and transmit raw MIDI data one byte at a time, without analyzing the MIDI messages at all. These objects can then be connected to objects that record the MIDI data, process it, or play it back.

You will probably end up using the specialized MIDI objects the most. They filter the raw MIDI data coming into Max, and output only the vital information. For example, the **notein** object looks only for MIDI note-on messages, and when it receives what it is looking for, it outputs the key number, the velocity, and the channel number. Similarly, the **bendin** object looks only for incoming pitch bend messages, and sends out the amount of pitch bend and the channel number. Other specialized MIDI input objects are **ctlin** for control change data, **polyin** for polyphonic key pressure data, **touchin** for aftertouch data, **pgmin** for program change data, **rtin** for real time messages, and **sysexin** for system exclusive messages.

Each of the specialized MIDI objects that receives channel voice messages has a transmitting counterpart, which receives numbers from other Max objects and uses those numbers as data for transmitting specific MIDI channel voice messages. The specialized MIDI transmitting objects are: **noteout**, **polyout**, **ctlout**, **pgmout**, **touchout**, and **bendout**.

Processing MIDI

Once MIDI data has been received by the MIDI objects, it can be modified and manipulated by other objects in Max. Max has many objects designed specifically to handle MIDI data.

The **midiparse** object receives numbers from **midiin**, analyzes the type of MIDI message, and routes MIDI channel voice data out specific outlets. The **seq** object is a simple MIDI

sequencer that records a sequence of data received from **midiin** and can play it back later, even changing the speed. A multi-track sequencing object, **mtr**, can record many separate tracks of data, then play them back either simultaneously or individually. The **detonate** object is a complex multi-track sequencer with a graphic editing window and score-following capability. To format numbers into well-formed MIDI messages, there is a **midiformat** object which prepares MIDI messages, then sends them to **midiout** to be transmitted out from Max.

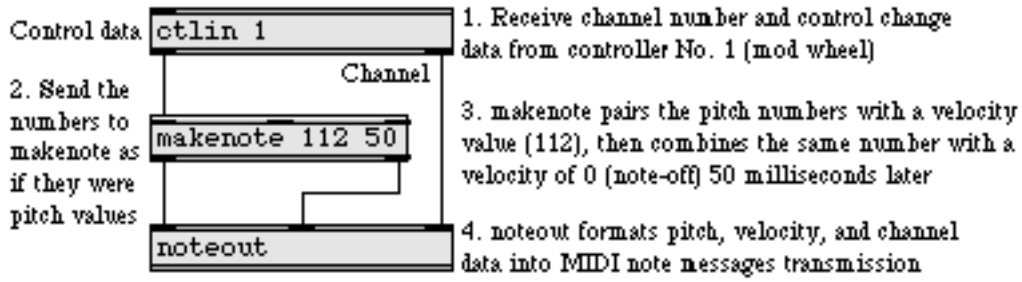
The **borax** object analyzes note-on data received from **notein**, and sends out information regarding the duration of notes, the time between note-ons, how many notes are being held at any given moment, how many notes have been played, etc. Both **borax** and the **poly** object are capable of assigning voice numbers to the notes that pass through them, and **poly** can steal voices and provide note-off messages to turn off notes that are being held.

The **stripnote** object filters out note-off messages (note-on messages with a velocity of 0) received from **notein**, passing on only note-on messages with a positive velocity. The **makenote** object is like the reverse of **stripnote**. It adds note-offs after note-on messages generated within Max, then sends both the note-ons and the note-offs to **noteout** for transmission. To help avoid stuck notes, the objects **bag**, **flush**, and **midiflush** keep track of the note-ons they have received, and can be used to turn off any notes for which they have not yet received note-offs.

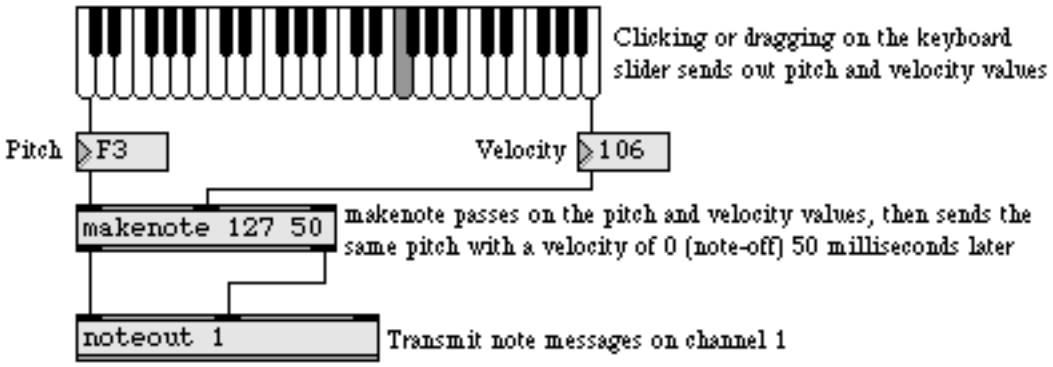
Max usually uses only 7-bit values (numbers from 0 to 127) for pitch bend data, and uses note-on messages with a velocity of 0 to express a note-off. Max is capable, however, of recognizing and transmitting 14-bit precision pitch bend values and MIDI note-off messages with release velocity using **xbendin**, **xbendout**, **xnotein**, and **xnoteout**.

The above objects are specifically designed to handle MIDI data, but since all MIDI values are just expressed as ints in Max, they can be processed in any way imaginable. The **pipe** object, for instance, delays all numbers passing through it by a certain amount of time. Or the **+** object can be used to add some number to a MIDI pitch value, thus transposing the pitch. The other sections in the manuals will show you a few of the many ways to process numbers in Max.

MIDI data can be reassigned to have other meanings. For example, data from the mod wheel of the MIDI keyboard could be received with **ctlin**, sent to **makenote** (to combine the number with a velocity value and add a note-off after it), then transmitted by **noteout**, as shown below. This would let you play *notes* with the mod wheel.

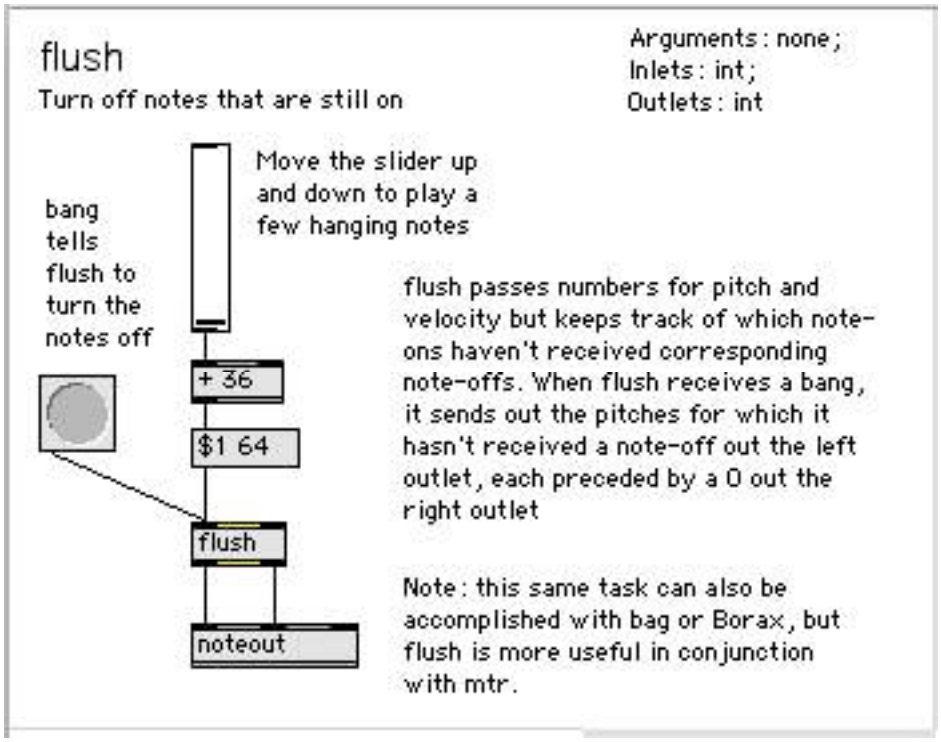


Numbers can also be generated within Max, either automatically with timing objects such as **metro**, **tempo**, and **clocker**, or interactively onscreen with user interface objects such as **slider**, **dial**, and **number box**. The numbers generated in Max can be sent to MIDI objects and transmitted out to play music.



Help

You can see Help files for built-in and external Max objects at any time. Option-clicking on Macintosh or Alt-clicking on Windows on an object in an unlocked Patcher opens the Help file for that object. You can also Option-click on Macintosh or Alt-click on Windows on any object's box in a locked Patcher to open a help file for that object if **Help from Locked Patchers** in the Options menu is checked. Help files are actual operating patches that let you see an object "in action." The following example shows a typical Help file.



In an unlocked Patcher, you can also open the Help file for an object by selecting the object and then choosing **Help** from the Help menu. You can open Help files from the New Object List by holding down the Option key on Macintosh or the Alt key on Windows while you select the object's name from the list.

If you still have questions about an object, look up the object's complete description in the Objects section of the **Max Reference Manual**. For help with other Max topics, consult the other chapters of that manual, or look up your topic in the **Max Object Thesaurus** or the Index.

File Menu

New... This set of pulldown menus lets you create new Max windows. The submenu options are described below:

- **Patcher** Create a new Patcher window.
- **Table** Create a new Table window, to make a **table** by drawing in values.
- **Text** Create a new Text window, to enter values in a text file, or simply to use as a note pad. Often values can be entered most efficiently by typing them in as text. For example, a **table** file can be created just by typing the word table, followed by a space-separated list of values. Once such a text file has been saved, it can be read by **table** objects in a patch. A **funbuff** file can be created by typing the word funbuff, followed by a space-separated list of alternating x and y values. Objects for which you might want to type in data in a Text window include **coll**, **env**, **funbuff**, **lib**, **mtr**, **seq**, **table**, and **Text**.
- **Timeline** Create a new Timeline window, for creating a graphic score of Max messages.

After Patcher, Table, Text, and Timeline, a list of *templates* appears in the New menu. Templates are Max patcher documents that open in untitled windows of preconfigured sizes. They can include objects to get you started with a certain type of application. To create your own template, simple save any patcher file in the **templates** folder inside the patches folder inside the Max application folder.

Open… Open an existing Max document. Max automatically determines the proper type of window in which to display the document—Patcher, Table, Text, or Timeline.

Open Recent Displays a submenu of opened and save files to save you time in opening a file you have work with recently.

Open as Text — Force Max to open an existing document in a Text window for editing. Any Max document or plain text file can be opened with this command.

Close — Close the active (foreground) window. If changes have been made to the document since it was last saved, Max will ask you if you want to save those changes.

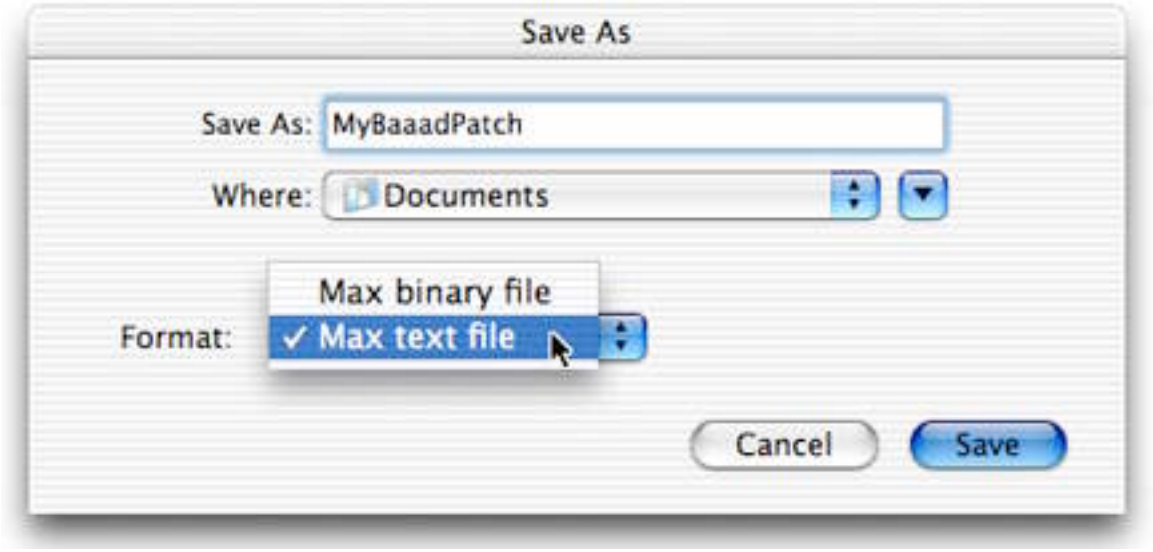
Save — Save the active document. The values stored in **number box** objects and other storage locations are not saved with a Patcher document. Max doesn't save the values that are stored in data objects such as **table**, **coll**, or **funbuff** within a Patcher unless the *Save With Patcher* option is set for those objects (although you can use the **pattr** family of objects to save the state of Max patches).

When you save an Untitled document, Max automatically uses the **Save As…** command, so that you can rename the document.

When saving a patch for the first time, it is possible to protect it from future modification, by holding down the Option key on Macintosh or the Alt key on Windows while clicking on the *Save* button in the dialog box. This renders the file uneditable when it is opened in the future. (Before you close the file, however, you can still edit it and save it again without this edit protection.)

Save As… — Save the active document in a separate file with a different name. The document is saved under the new name, and the document with the old name is closed unchanged.

By default, most files are saved in Max Binary format, which loads most quickly and takes up the least disk space. You can also save a file in Text format. A pop-up menu in the standard Save As dialog box allows you to choose.



Build Collective / Application / Plug-In ...
Save the active document, along with all other files necessary for it to function as a single collective file, plug-in, or standalone application. For more information on the collective save dialog, see the Collectives topic.

Midi Setup...
Configure Max's MIDI connection to the outside world. The MIDI Setup dialog box is presented to you the first time you use Max. From then on, Max remembers your setup in the *Max Preferences* file, and are recalled each time the Max application is opened.

You should assign your various MIDI devices to specific ports using the Audio MIDI Setup application. The MIDI Setup dialog in Max then allows you to give each port an abbreviation (a letter a-z), and assign a range of MIDI channels to each input and output port.

If you click on the *Auto Setup* button, Max will automatically assign IDs and channel offsets in a reasonable configuration.

Page Setup...
Specify paper size, paper orientation, and printing options before printing. If the patch you wish to print exceeds the bounds of the paper, you can reduce the image by specifying a percentage of reduction.

Print...
Print the contents of the active Patcher or Text window.

Install...
Install Max external objects. When the Max application is opened, it installs external objects in the max-startup folder. Other external objects can be installed later with **Install...**. However, if an external object is in Max's search path, you can just type its name into an object box to install it.

Quit
(Windows only in File menu. On Macintosh, the Quit item is in the

Application menu.) Quit Max. If you have any unsaved windows, you'll be asked if you wish to save changes to the windows.

Edit Menu

Undo — Undo the most recent editing change in an unlocked Patcher or Text window. Editing changes that can be undone in a Patcher window are **Undo**, **Cut**, **Copy**, **Clear**, **Paste Replace**, **Fix Width**, **Align**, **Hide on Lock**, **Show on Lock**, **Bring to Front**, **Send to Back**, **Ignore Click**, **Respond to Click**, **Include in Background**, **Exclude from Background**, changing an object's font and size, and moving an object.

Cut — In a unlocked Patcher window, table editing window, or Timeline window, **Cut** removes the selection and copies it to the clipboard

When **Cut** is chosen when the Max window is active, the text contents of the Max window is erased and text is copied to the clipboard.

Copy — In a unlocked Patcher window, table editing window, or Timeline window, **Copy** copies the selection to the clipboard without deleting it.

When **Copy** is chosen when the Max window is active, the text contents of the Max window is copied to the clipboard.

Paste — In a unlocked Patcher window, table editing window, or Timeline window, paste the contents of the clipboard into the active window. When you move objects from one Patcher window to another, Max will paste the objects into the new window in the same position they had in the old window.

Clear — In a unlocked Patcher window, table editing window, or Timeline window, **Clear** deletes the selection without copying it to the clipboard. Pressing the Delete key on Macintosh or Backspace key on Windows has the same effect in most Max windows.

Duplicate — In an unlocked Patcher or Timeline window, **Duplicate** makes a copy of the selection. The duplicated objects are placed just below and to the right of the original objects, and are automatically highlighted so that they can be dragged to the desired location. After the duplicate objects have been dragged, Max takes note of their position relative to the original objects. If you then reduplicate the objects, without deselecting them, the new duplicates will be placed in the same position relative to the selection.

In an unlocked Patcher window, you can also duplicate quickly by Option-dragging on an object on Macintosh, or Alt-dragging on Windows, which creates a copy of the object and lets you drag it away from the original.

Select All — In an unlocked Patcher window, table editing window, Text window, or Timeline window, this command selects all objects, values, or text that are not hidden (i.e., It does not select foreground boxes if the foreground is hidden. It does not select background boxes if the background is hidden. It does not select patch cords if connections are hidden). In an unlocked Patcher window, **Select All** works differently from selecting all the objects by dragging around them with the mouse, which selects the objects but not the patch cords. The distinction is important if you want to hide all the selected objects and patch cords by choosing **Hide On Lock** from the Object menu. You can select everything with the **Select All** command, then exclude individual items from the selection by holding down the Shift key and clicking on them.

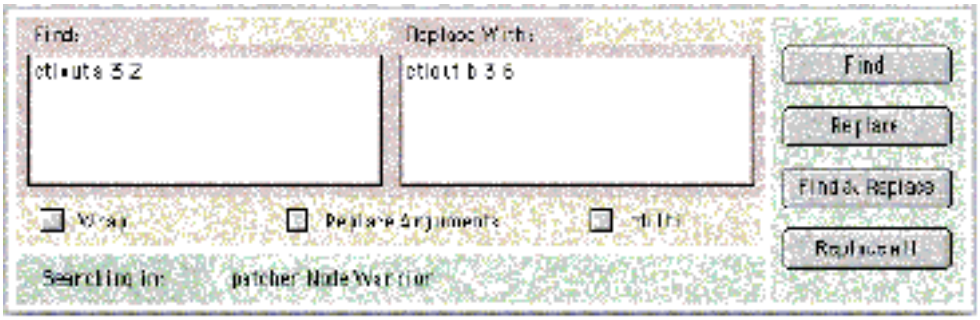
Paste Picture — Paste a graphic image into a Patcher window from the clipboard. Graphics can be made functional by placing the transparent button object, **ubutton**, over the image. If you have copied objects in a Patcher window to the clipboard, **Paste Picture** pastes an image of the objects, rather than the objects themselves. **Paste Picture** creates an object called **vpicture** in the Patcher window. **vpicture** has no messages, arguments, or output. For more flexible picture handling, save the picture as a file and load it into the **fpic** object.

Paste Replace — Replace the currently selected objects in an unlocked Patcher with the object on the clipboard.This command is only enabled if you have copied a single item to the clipboard. It can be very useful for updating the settings of a group of objects.

Encapsulate — Create a subpatcher containing the selected objects in a Patcher. For more information, see the Encapsulation topic in the Max Tutorials & Topics manual.

De-Encapsulate — If a subpatcher is selected in a Patcher, De-Encapsulate will replace the subpatcher with the objects it contains. For more information, see the Encapsulation topic in the Max Tutorials & Topics manual.

Find… — Open a window for finding a string of characters in the active window. **Find…** searches for a certain word or words and optionally replaces them with other words. This is especially useful in either Text windows or Patcher windows that contain many instances of the same word or words. In Patcher windows, Find operates on object boxes, message boxes, and comments.



The Find dialog has two main text entry fields, one for the text you would like to search for, and one for the text you would like to replace it with. The checkboxes operate as follows:

If the *Wrap* option is checked, Max returns to the beginning of the Text or Patcher window after it reaches the end, and searches until it is back where it started.

When the *Replace Arguments* option is checked, and the search finds an object, message, or comment box that begins with the text in the Find text field, the Replace string replaces the entire contents of the text object. When *Replace Arguments* is not checked, the replace string will replace only the number of arguments in the replace string itself, leaving the rest unchanged.

The *Replace Arguments* option has no effect in a Text window.

When the *Multi* option is checked, Max searches through all open Patcher windows after exhausting the search in the Patcher window where you started your search.

The *Find* button performs a search on the text in the Find field, but merely selects the text found, if any.

The *Replace* button replaces the selected text with the text in the Replace text field.

The *Find & Replace* button performs a search on the text in the Find field, and, if found, replaces it with the text in the Replace field.

The *Replace All* button replaces all instances of the text in the Find field with the text in the Replace field.

Find Next — Repeat the most recent search performed with **Find…**, finding the next instance of the search string.

Replace — Replace the selected text with the replace string from the Find… dialog box

Replace and Find — Replace the selected text with the text in the Replace field of the Find window, then perform another search for the text in the Find field of the Find window.

Replace All — Replace all instances of the text in the Find field of the Find window with the text in the Replace field of the Find window.

Resume — It is possible to get Max working so hard and fast that it doesn't have time to respond to your commands. For example, if you have a number of timing objects such as **metro** sending out messages as fast as they can, or if you have created a loop with little or no delay between messages, Max may be too busy to pay attention to you. Holding down the Command key on Macintosh or the Control key on Windows and typing a period stops Max's scheduler, giving you time to turn off some of the overloading processes. Choose **Resume** to restart Max's scheduler.

If you have a bug in your program that causes a Stack Overflow error, such as connecting the outlet of an object in such a way that it indirectly sends input back into its own triggering inlet, Max will stop its scheduler and notify you of the error. After you have fixed the bug, choose **Resume** to restart Max's scheduler

View Menu

Edit — When **Edit** is checked, the active Patcher window can be edited. When **Edit** is unchecked, the active Patcher window is *locked*, and its user interface can be operated. Locking and unlocking the active window by holding down the Command key on Macintosh or Control key on Windows while clicking in white space within the window has the same effect as choosing **Edit** from the View menu. Macintosh versions also include a transparent rectangular pill on the right side of the window that can be used to toggle between locked and unlocked Patcher states.

Hide Object Palette — Hide the palette of objects at the top of an unlocked patcher window. When the object palette is hidden, this menu item changes to **Show Object Palette**, which when chosen, makes the object palette reappear.

Hide Connections — Hide all patch cords in an unlocked patcher window. **Hide Connections** also unselects any selected patch cords so that they can not be deleted while connections are hidden. When all patch cords are hidden, this menu item changes to **Show Connections**, which when chosen, makes all patch cords reappear. Since any patch cords that are not marked as hidden reappear when you lock the patcher, the purpose of this command is to move patch cords out of the way for precise editing or positioning of objects.

Hide Imageburgers — Hide any images associated with objects (known as *imageburgers*), restoring the standard object box text view. To set an object's image, select the object and choose **Imageburger...** from the Object menu. Then use the Object Image window that appears to assign a picture file to the object. When imageburgers are hidden, this menu command changes to **Show Imageburgers**, which when chosen, makes object images reappear.

Hide Foreground — Hide objects that belong to the foreground layer in a Patcher window. All objects in a Patcher are by default in the foreground layer unless they are included in the background by choosing **Include in Background** from the Object menu. This command hides only objects, not patch cords. It is intended to make it easier to edit or position objects in the background layer. When foreground objects are hidden, this menu command changes to **Show Foreground**, which when chosen, makes the foreground objects reappear. Note that if hidden, foreground objects will not reappear when the Patcher window is locked.

Hide Background — Hide objects that belong to the background layer in a Patcher window. All objects in a Patcher are by default in the foreground layer unless they are included in the background by choosing **Include in Background** from the

Object menu. This command hides only objects, not patch cords. It is intended to make it easier to edit or position objects in the foreground layer. When background objects are hidden, this menu command changes to **Show Background**, which when chosen, makes the background objects reappear. Note that if hidden, background objects will not reappear when the Patcher window is locked

Lock Background — Make it impossible to select or edit objects belonging to the background layer in an unlocked Patcher window. This is useful for editing over an image in the background while viewing the image. After choosing **Lock Background**, the menu item is checked. Choosing **Lock Background** when the item is checked unlocks the background layer and makes the check mark disappear from the menu item.

Set Origin — This item is enabled if you have scrolled a Patcher window so that its original top- left corner is no longer in the same location (it might be hidden or somewhere in the middle of the screen). Choosing **Set Origin** in an unlocked Patcher window defines the current top-left corner as the one that will appear when the patcher window is re-opened after being saved.

Restore Origins — This item is enabled if an origin has been defined with **Set Origin** and you have scrolled the Patcher window away from it. **Restore Origin** scrolls the Patcher window to make the defined origin the top-left corner.

Object Menu

Fix Width — Adjust the width of an object box, a **comment**, or a **message** box to accommodate all the text it contains on a single line. When **Auto Fix Width** is checked in the Options menu, Max will automatically adjust the width of object and **message** boxes each time they are edited.

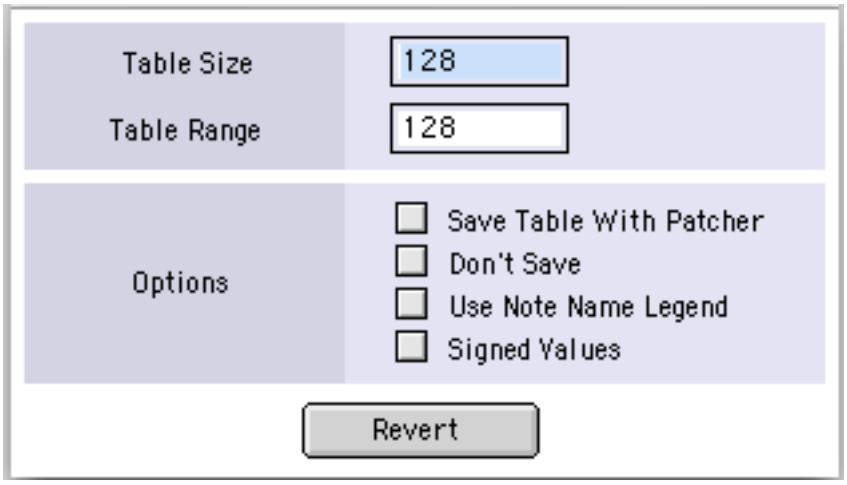
Align — If two or more objects are select, this command adjusts the position of all selected objects so that they are perfectly aligned vertically or horizontally. All selected objects are aligned in relation to the leftmost object in the selection.

The Align feature makes an educated guess as to whether to align the selected objects vertically or horizontally. If all objects in the selection are below the top edge of the leftmost object, but not completely to the right of the right edge of the leftmost object, the objects are aligned vertically. Otherwise, they are aligned horizontally.

If a patch cord is selected, Align squares off the patch cord, attempting to avoid crossing over boxes in doing so. In certain cases, the task of avoiding

boxes is too difficult and **Align** gives up without changing the patch cord's shape.

Get Info... Opens an Inspector window that lets you change an object's properties. Almost all of the user interface objects, as well as **table** objects, have certain characteristics (such as size and range) which can be set with **Get Info…**. The actual contents of the Inspector window will depend on the object.



Inspector window for a **table**

Choosing **Get Info…** in an unlocked Patcher window prints a list in the Max window of all subpatches and external objects used in that patch. You can use this to locate the external object or subpatch files.

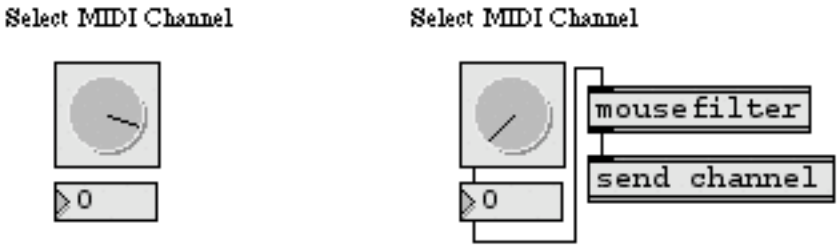
Color This submenu features 16 preset colors that can be applied to a number of user interface objects and patch cords. You can change the colors by choosing **Edit Colors…** in the Options menu.

Name... Assigns a variable name for an object in a patcher window that can be used in a script. This name is not the same as a name for a **table** or **coll** object; it is local to a Patcher window and can only be used with script messages to the **thispatcher** object. For more information, see the Scripting tutorials in the **Tutorials and Topic**s manual.

Imageburger... Replaces an instance of a Max object or patcher with a graphic image. To assign a graphic image, select an object and choose this menu item. A dialog box will appear that allows you to choose the graphic you want to use.

Hide on Lock

Make the selected objects and patch cords invisible when the Patcher window is locked, which reduces the cluttered appearance in a patch. Note that user inter- face objects that are hidden will not respond to mouse clicks.



Some objects and cords hidden *Nothing hidden*

Show on Lock

Set the selected objects and patch cords to be visible when the Patcher window is locked, undoing the action of **Hide On Lock**.

Bring to Front

Bring the selected objects to the front of their current layer within the Patcher window. If a user interface object such as a **dial** or a **ubutton** is partially obscured by another object (or a picture) on top of it, **Bring to Front** puts it in the fore- ground and allows it to respond normally to mouse clicks. When a new object is created, it is automatically made the frontmost object

Send to Back

Send the selected objects to the back of their current layer of the Patcher window, behind all other objects. If an object sent to the back was covering any user interface objects, they will now respond normally to mouse clicks.

Ignore Click

Cause the selected user interface object(s) not to respond to mouse clicks when the Patcher is locked. This attribute stays with any copies subsequently made of the object with the **Cut**, **Copy**, or **Duplicate** commands in the File menu. If an object is set to **Ignore Click**, and there is another user interface object behind that one, the background object will receive the mouse click.

Respond to Click

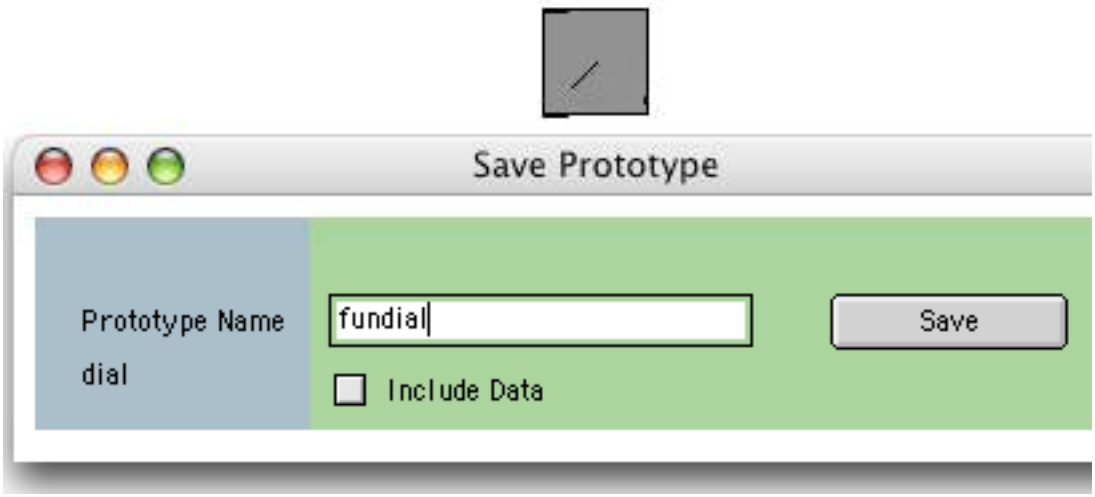
Undo the effect of a prior **Ignore Click** command, restoring the selected user interface object to its normal way of responding to mouse clicks.

Include in Background

Place the selected objects in the background layer. Objects in the background layer appear behind objects in the foreground layer, even those on which the **Send to Back** command has been used. In addition, the background layer can be locked so that foreground objects can be easily edited on top of background objects.

Remove from Background — Remove the selected objects from the background layer

Save Prototype... — Open a window for entering a name to save the settings of the selected object in a package called a Prototype. Prototypes are useful for saving combinations of user interface object settings you plan to use often. For example, you can save color combinations you like, or complex image file configurations used by picture-based controls.



To save a prototype, type in a name, then click the Save button. The prototype will then appear in the **Prototypes** submenu of the Object menu (see below). Prototypes are also listed in a pop-up menu you obtain when holding down the mouse button when creating a new object.

Check the box labeled Include Data in the Save Prototype window if you want a preset containing the object's data to be saved with the prototype. Typically, you will not want to include the preset in a prototype.

Prototypes — The **Prototypes** submenu of the Object menu will be enabled if there are prototypes saved for the type of object that is currently selected. Prototypes are named combinations of settings (such as colors and image files) you can apply to user interface objects. For example, if a dial is selected, prototypes for dials will be shown.

Choose a prototype by name from the Prototypes submenu to apply the settings of the selected object with the prototype settings. Currently, any data stored in the object (such as the current value of a dial) is lost when you apply a prototype.

Font Menu

Each Patcher or Text window, and each object that displays text in a Patcher window, can be assigned a specific font and font size.

To change the font of an object, select it, then choose the desired font characteristics from the Font menu. When no objects are selected, choosing a font or font size will set the default font characteristics for new objects created in the active window. In a locked Patcher window, you can make a textedit object active and change its font with this menu.

Choosing a font or font size when the Max window is active changes the font used in the Max window and also sets the default font characteristics for any new Patcher or Text window created from then on, and stores those settings in the *Max Preferences* file.

9-36 Specify the font size. Font sizes available on your system for the currently chosen font will be shown in outline.

(Fonts) The fonts shown in the menu depend on the fonts available on your system. On Windows, fonts are listed in the Faces submenu.

Options Menu

Settings for all the commands in the Options menu are stored in the *Max Preferences* file. This file is located in the /Library/Preferences/Max 4 Preferences Folder folder on Macintosh. On Windows, this folder is called "Max 4 Preferences FIles", and its location varies according to the name of the user logged in (e.g. C:\Documents and Settings\<user>\Application Data\Cycling '74\Max 4 Preferences Files\). Preferences are recalled each time the Max application is opened.

Overdrive When **Overdrive** is enabled, Max gives priority to timing and MIDI processing over screen drawing and user interface tasks such as responding to mouse clicks. This will often yield more accurate event timing, but may preempt screen drawing tasks. (Note: The **Overdrive** command and the **Enable** command in the Trace menu are mutually exclusive.)

All Windows Active

Allow any window to respond to the mouse, without first being brought to the front. Normally (when **All Windows Active** is not enabled), only objects in the frontmost window can respond to mouse clicks, and clicking on a window in the background brings it to the foreground. With **All Windows Active** checked, objects in a background window will respond to the mouse, but you must click in the title bar of a background window (or choose its name from the Windows menu) to bring it to the foreground.

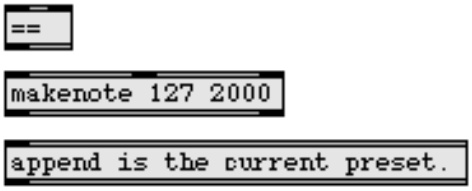
Regardless of the setting of **All Windows Active**, you can only type numbers into a **number box** if it is in the foreground window. Keystrokes on the computer keyboard will be sent out of the outlets of **key** and **keyup** objects in any loaded patch or subpatch, regardless of the setting of the **All Windows Active** option.

New Object List

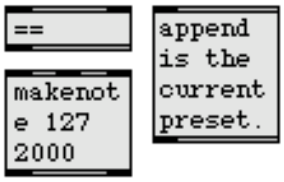
When **New Object List** is enabled, Max shows a complete list of available objects whenever you create a new object box. To override seeing the New Object List window temporarily, hold down the Option key on Macintosh or the Alt key on Windows when placing a new object box in the Patcher window. See the *Objects* chapter of this manual for details about the New Object List window.

Auto Fix Width

When **Auto Fix Width** is enabled, the width of an object or **message** box is automatically adjusted each time you edit the text inside it. The box is changed so that all the text fits on a single line. You can do this to any selected text box manually by choosing **Fix Width** from the Object menu. Some examples of **Fix Width** are shown below.



Auto Fix Width on



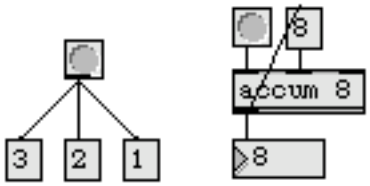
Auto Fix Width off

Width fixed by hand

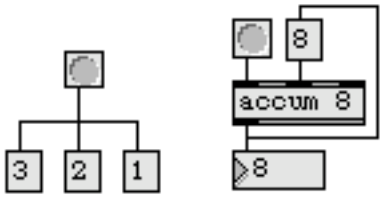
Segmented Patch Cords

When **Segmented Patch Cords** is *not* checked, you can draw a patch cord between two objects simply by dragging with the mouse from the outlet of one object to the inlet of another object.

The procedure for drawing patch cords differs slightly when **Segmented Patch Cords** is checked. First you click on the outlet of one object, then click at each of the points where you want the patch cord to bend, then click on the inlet of the other object. The shape of a segmented patch cord can be adjusted afterward by dragging on one of the segments with the mouse.



Straight patch cords



Segmented patch cords

If you make a mistake while drawing a segmented patch cord, you can get rid of the most recent segment by Option-clicking on Macintosh or Alt-clicking on Windows with the mouse. Or you can get rid of the whole patch cord by Command-clicking anywhere on Macintosh or Control-clicking anywhere on Windows.

Although segmented patch cords redraw a little more slowly when selected or deselected, they function exactly like straight patch cords, and do not take up additional memory when the patch is loaded.

To make a segmented patch cord when **Segmented Patch Cords** is not checked, hold the shift key down when clicking on an outlet. Similarly, to make a straight patch cord when **Segmented Patch Cords** is checked, hold the shift key down when clicking on an outlet.

Assistance

When **Assistance** is enabled, the Patcher window gives you a running commentary about the function of each object's inlets and outlets when the mouse moves over an inlet or outlet. The following example shows **Assistance** for the right inlet of the **accum** object.

When you edit a patch to be used as an object inside other patches, you can type in your own Assistance messages for your object's inlets and outlets. Select an **inlet** or **outlet** object in your patch, and choose **Get Info…** from the Object menu. You will be presented with a dialog box which lets you type in a description of that **inlet** or **outlet**. When you use your object inside another patch, the Assistance messages will appear when the mouse is placed on the object box's inlets or outlets.

Float Display Correction

The way a computer stores floating-point numbers means that not all numbers can be represented. This means that you will often see values such as 2.3 displayed as 2.29999. When **Float Display Correction** is checked, Max rounds numbers and truncates them at what it guesses is the least significant digit.

Max Window at Startup

Show the Max window when the Max application is opened. Unchecking this option keeps it hidden until you explicitly open it by choosing **Max** from the Windows menu.

Enhanced File Preview

(Macintosh only) When checked, this option asks QuickTime to preview all files in the standard open file dialog, even those for which preview information does not exist. This may mean significant delays when creating previews for large graphics files. Uncheck this option if you are working with many large graphics files that do not contain preview information.

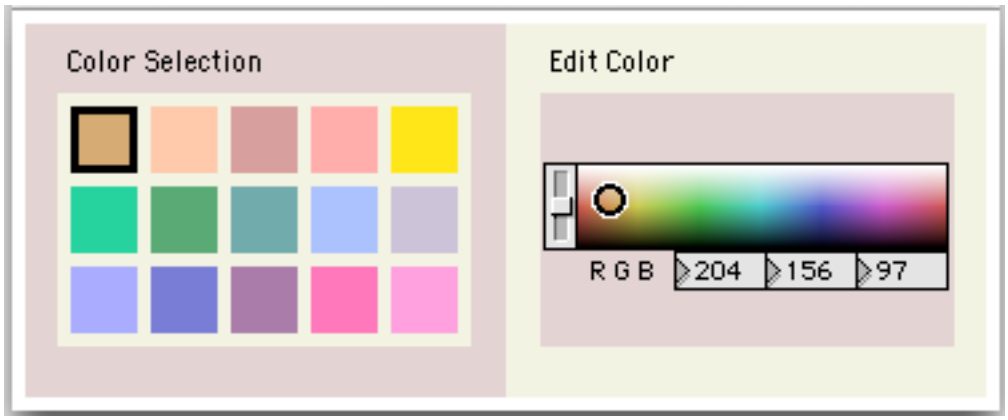
Save Dialog

When you choose Save As... or use Max object's save file dialog to specify the

location of a file, the last place you visited is typically shown by default. The Save Dialog Shows File Location option, when checked, shows the directory containing the file you are going to save *if it has already been saved*. If the file you are going to save has not yet been saved, the setting of Save Dialog Shows File Location has no effect.

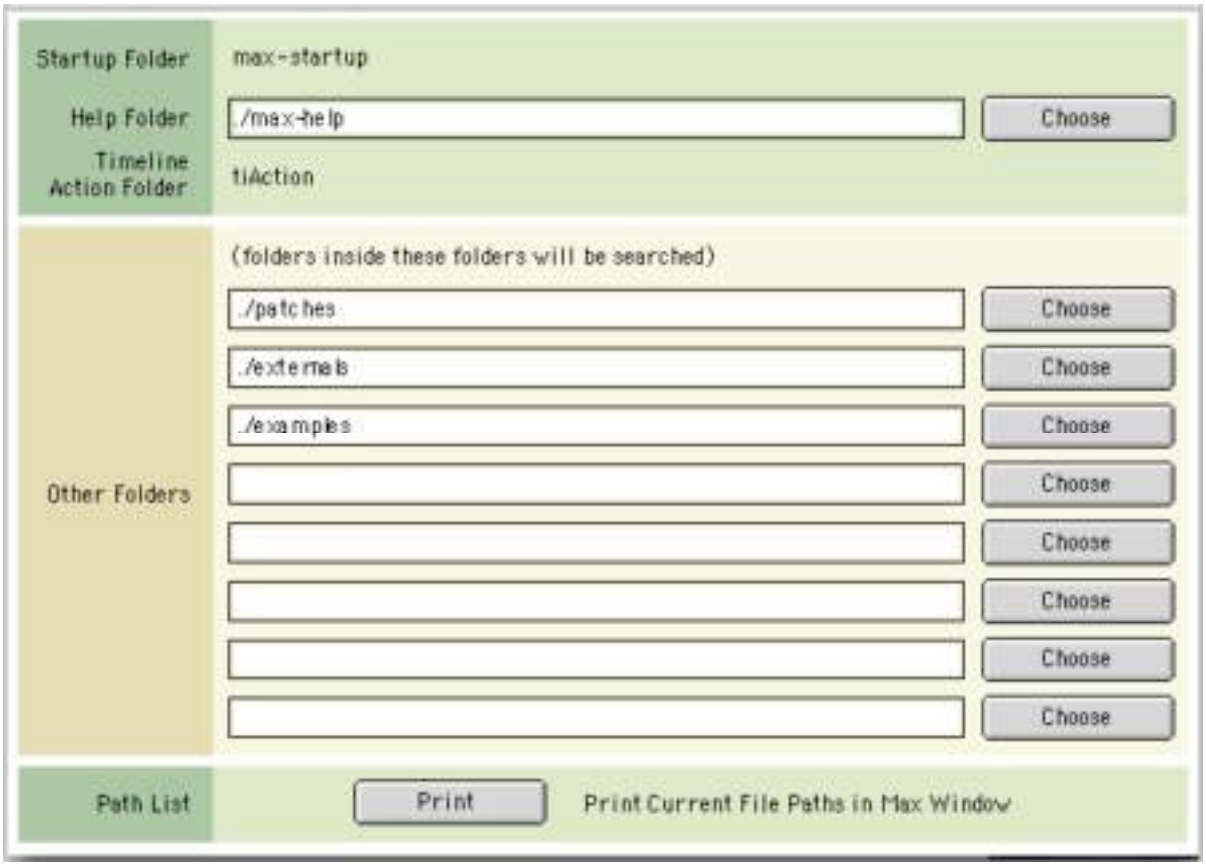
DSP Status... If MSP is installed, opens the DSP status window for changing Max/MSP audio settings. See the Audio Input and Output chapter of the MSP **Getting Started** manual for more information.

Colors... You can apply color to a number of Max objects and patch cords using the Color submenu of the Object menu. The **Colors...** command opens a window that edits the palette of colors used in this menu.



To edit a color, select one of the fifteen colors on the palette. You will see the selected color displayed on the right using the color swatch, and you will see an RGB color listing for the selected color. Change the selected color by either moving the swatch cursor around in the color space or by typing the RGB value for the color you want. Object colors in all open patchers are updated when you close the window.

File Preferences... Opens a window for setting the Max *file search path:* the folders that will be searched when Max looks for documents by name.



When the Max application is opened, it loads all external objects and patches contained in the max-startup folder.

The *Help Folder* is where online help files for each object are stored. When you Option-click on Macintosh or Alt-click on Windows on an object, Max searches the help folder for a file with that name and the .help suffix.

The eight slots listed for Other Folders comprise the root levels of the Max search path. You can either type the names of the folders you want to include, or use the Choose button to use a folder selection dialog to specify it.

If a dot-slash (./) precedes the folder name listed in the File Preferences window, it is a *subfolder* of the folder in which the Max application resides. To specify a folder elsewhere, you must type in the full path name of the folder, beginning with the name of the hard disk that contains it. Use slashes to specify folders within folders. For example, a subfolder called *InProgress* in the *MyPatches* folder in the Max application folder would be specified as ./MyPatches/InProgress. If a name does not begin with a dot-slash, it must be the name of a volume (a disk or disk partition), and any folders in the volume are separated by slashes. For example, if you want Max to look in a folder called

New Compositions inside the folder called *MIDI files* on the disk named *MyDrive*, specify it as MyDrive:/MIDI files/New Compositions.

If there are any spaces in the name when you type it in, you need to enclose the entire name in double quotes.

Max's *file search path* is the list of folders that will be searched when Max looks for documents.

The Cycling '74 folder—a folder that serves as a global repository for plug-in versions of Max used within other application or multiple versions of Max that you may be using—is searched *before* any of the locations specified in the search path. It is located in the Drive:/Library/Application Support folder on Macintosh or in the C:\Program Files\Common Files folder on Windows. The file search path proceeds as follows:

1. the folder that contains the most recently loaded patch (including a patch that is in the process of being loaded),

2. the folders specified in the **File Preferences** dialog,

3. the folder that contains the Max application

Performance Options...

Opens a window that displays a series of parameters that will allow you to fine-tune Max event scheduling, audio, and graphics performance to suit your needs. Click the About... buttons next to each option to learn more about how it will impact performance.

Text Selection...

Opens a window that displays and explains options for selecting text in a patcher window. Each option setting is described by a Quicktime movie showing how the Patcher behaves.

Select Text on Click, when enabled, immediately enters edit mode when you click on an object box, message box, or comment box. It visually selects the text in the box. If Select Text on Click is disabled, the entire box is selected, which facilitates non-textual editing operations (such as moving and copying). Clicking inside an entirely selected box and immediately releasing the mouse button will enter edit mode for that box.

Typing Automatically Edits Selected Box is typically on only if Select Text on Click is off. If the entire box is selected, typing immediately enters edit mode. If you turn this option off, a text object must be in edit mode before typing will affect the text inside the box.

Trace Menu

Enable/ Disable

Enable *Trace* mode, which allows you to step through every single message sent in a patch, is used for debugging purposes. If Trace mode is currently enabled, choosing this command will disable it. Choosing the **Enable** command automatically disables **Overdrive** if it is currently checked in the Options menu, before enabling the Trace mode for debugging.

Once Trace is enabled, the next message sent out an outlet will cause the patch cord through which it's travelling to blink, and information about the message will be printed in the Max window. You can then use other commands in the Trace menu to continue the debugging process. When you **Disable** a Trace in progress, Max finishes tracing the current messages, then reverts to normal operating mode.

Step

Send the message that's in the blinking patch cord into its destination inlet, and advance to the resulting next message. The next message will be reported in the Max window, and the patch cord through which it is travelling will blink.

Continue

Advance in normal operating mode until the next breakpoint is encountered. (See the **Set Breakpoint** command, below.)

Abort

Immediately stop the execution of the patch.

Auto Step

Step through the messages in the patch at a steady, moderate pace until a) the Trace is completed, b) a breakpoint is encountered, or c) a **Disable, Abort**, or **Auto Step** command is chosen from the Trace menu.

Set Breakpoint

With the Patcher window unlocked and a patch cord selected, **Set Breakpoint** will put a stopping point at that patch cord, at which a Trace will stop when it is operating on a **Continue** or **Auto Step** command.

Clear Breakpoint

With the Patcher window unlocked and a patch cord selected, **Clear Breakpoint** will remove from that patch cord any breakpoint previously set with the **Set Breakpoint** command.

Clear All Breakpoints

Remove all breakpoints from all loaded patches.

Window Menu

Close
Close All
Next
Previous
Cascade
Tile Horizontally
Tile Vertically

(Windows only) These standard Windows-specific menu items are used to manage multiple windows, and behave as they do in other Windows applications.

Hide Subwindows

Double-clicking on a **table** or **patcher** object, or on an object that is actually a subpatch, opens a *subwindow* to display the contents of that object. The **Hide Subwindows** command closes any subwindows that may be open.

Send Window to Back

Send the currently active window to the background, behind all other open windows.

Max

Bring the Max window to the front, or show it if it has been hidden.

Show Floating Inspector

Opens a floating window that changes its contents to the Inspector for the currently selected object in the active unlocked Patcher window. Since Inspector windows only operate on one object at a time, the Floating Inspector window will hide its Inspector if you select multiple objects.

(other)

The names of all other open windows are listed in the Windows menu. Choosing a window name brings that window to the front, making it the active window. A diamond (_) on Macintosh or an asterisk on Windows appearing to the left of a window's name indicates that it has been modified since it was opened, and the changes have not yet been saved.

Choosing a window's name from the Window menu while holding down the Option key on Macintosh or the Alt key or Windows moves the window onto the main screen and resizes it to fit within the bounds of the main screen. You can use this if you open a file whose window is partially or completely off the screen.

Extras Menu

The Extras menu contains the names of patches that you can use for reference or utility purposes. You can add your own patches to this folder—choose **Adding Extras...** from the Extras menu for instructions. Note that patches opened using the Extras menu are brought to the front by choosing their name from the Extras menu. They are not listed in the Windows menu.

Adding Extras...	Gives you instructions for creating your own patches to the Extras menu.
Tips	Gives you a quick list of shortcuts you can use while working in a Patcher window.

In the Max/MSP installation, the following items are included in the Extras menu:

Audiotester	Lets you send a test signal out a desired channel.
Meterin	Displays an input signal level meter.
Meterout	Displays an output signal level meter.
MIDItester	Lets you test your current MIDI input and output setup.
Mousemeter	Gives you a quick readout of mouse position, and lets you measure distances relative to a position you can set.
Quickrecord	Lets you record your output to an audio file in the Max application folder.
Swatches	Lets you edit UI objects with multiple colors quickly.

Help Menu

Help… Choosing **Help…** when an object in an unlocked Patcher window is selected opens a help file describing that object. You can get help on any object at any time by holding down the Option key on Macintosh or the Alt key on Windows and clicking on its box. You can also Option-double click on Macintosh or Alt-double-click on its name in the New Object List window). You can also Option-click on Macintosh or Alt-click on Windows on any object's box in a locked Patcher to open a help file for that object if **Help from Locked Patchers** in the Options menu is checked.

Contextual Menus in the Patcher Window

When you Control-click inside a Patcher window on Macintosh or Right-click on in a Patcher on Windows, you'll get a pop-up menu that lists different items depending on what you click on. There are three basic menus: one for clicking on objects, one for clicking on patch cords, and one for clicking on blank space. In addition, Control-clicking on Macintosh or Right-clicking on Windows on the object box and message box icons in the Patcher window's palette display the Recent Object and Recent Message submenus described below for the Blank Space Contextual Menu.

Object Contextual Menu

This menu contains items from the Edit, Object, and Help menus. In addition, some objects, such as **bpatcher**, **js**, and **mxj**, add items to the bottom of this menu. See the manual page from the **Max Reference Manual** or the **MSP Reference Manual** for the object you selected for a description of the items it adds to this contextual menu.

Patch Cord Contextual Menu

This menu contains items from the Object menu that are applied to the currently selected patch cord.

Blank Space Contextual Menu

This menu contains items from the Object menu as well as a number of items specific to this context described below.

Object Palette This submenu displays all the icons for user interface objects in the Patcher window's object palette. Choosing one creates a new object of the type you

specify.

New Object — This submenu comprises a number of submenus that create a new object box and place some text in it. The organization and contents of the submenus is identical to the New Object List window that appears when you create an object box when **New Object List** is checked in the Options menu.

> **Note:** Using the Object Palette and New Object submenus allows you to create objects in a patcher while the Object Palette is hidden (choose **Hide Object Palette** from the View menu to do this). You may find you like this method of creating objects better than using the Object Palette.

Recent Object — This menu contains the text of the most recently created object boxes, in case you would like to create another copy of one of the boxes.

Recent Message — This menu contains the first part of the text of the most recently created message boxes, in case you would like to create another copy of one of the boxes.

Set Default Color — This submenu shows the Color submenu, allowing you to choose a color other than black for newly created patch cords and objects.

Paste From... — This submenu lists all of the available Clippings. A *Clipping* is a Patcher file that has been saved in the clippings folder inside the patches folder in the Max application folder. When you choose an item from this menu, the contents of the selected Patcher file will be pasted into the Patcher window, with the top-left corner of the window being moved to the point where you clicked to get the contextual menu.

In addition, an Other... menu item in the Paste From... submenu allows you to choose any Patcher file using an standard open file dialog and paste its contents into the current patcher window.

Object Quick Reference Menu

Option-Control-clicking on Macintosh or Alt-right-clicking on Windows on an object produces a pop-up menu that lists all of the messages you can send to the object along with some information about the arguments to those messages.

The message name is listed first, followed by the argument types (if any) in square brackets. If you see [int] after a message name, it means the message expects a single number as an argument. If you see [int, int, int] after a message name, it means the message expects three numbers as arguments. If you see [variable] after a message name, it means the argument can vary, and Max does not have enough information to determine the exact argument syntax required.

Beneath the message listing, there are two additional items. One opens the object's help file, and the other opens a file called boxquickref.help that explains the Object Quick Reference menu.

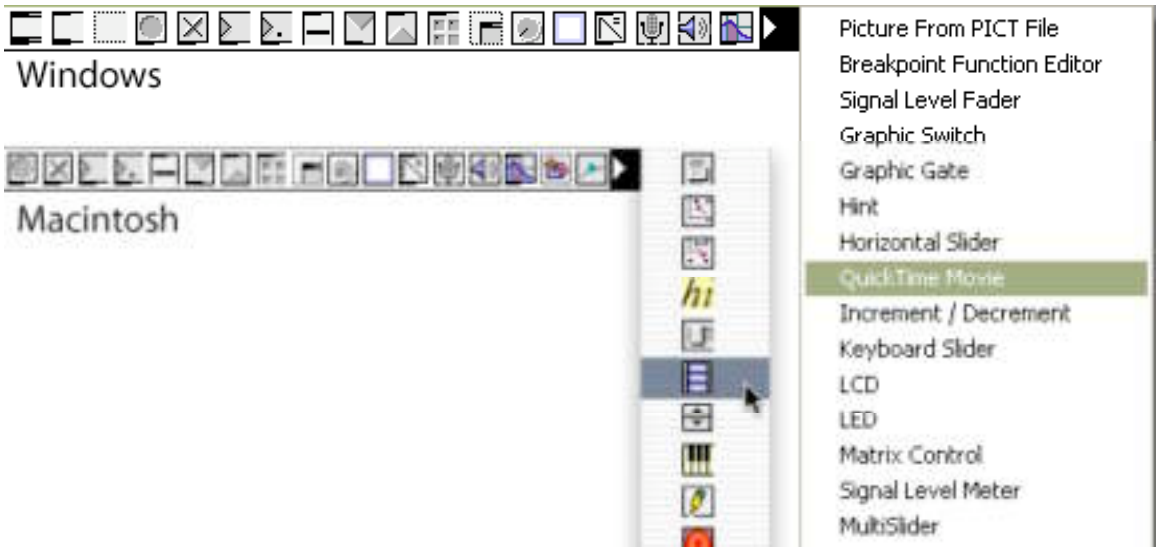
See Also

menu	Pop-up menu, to display text and send messages
menubar	Put up a custom menu bar
Objects	Creating a new object
Collectives	Grouping files to create a single application.

Object Palette

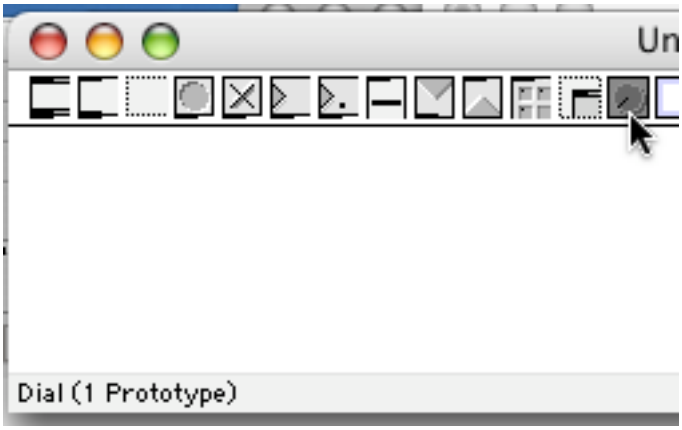
When the Patcher window is unlocked, the *object palette* appears at the top of the window. To create a new object in the Patcher window, simply click on the desired icon in the object palette (the cursor will change to that icon), then click where you want to place the object.

When the Patcher window is narrower than the width of the icons in the palette, the far right palette item in the window becomes an arrow. Holding down the mouse button on the arrow causes the rest of the object palette to be displayed as a pop-up menu. The items in the pop-up menu are displayed as icons on Macintosh or text on Windows. Choose the desired object from the menu. Note that as you move the mouse over different objects in the palette, they are described in the Assistance area of the Patcher window.



If you choose the wrong object by mistake, you can cancel by clicking on the blank area at the far left of the palette or pressing the Delete (Backspace) key.

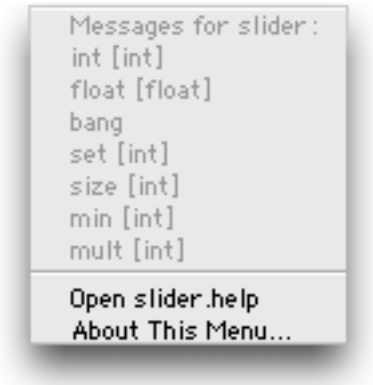
If an object in the palette has any prototypes you will see the number of prototypes after the object name listed in the assistance area of the Patcher. In the following example, the dial has one prototype.



To access the prototypes when creating an object via the object palette, hold the mouse button down momentarily. A pop-up menu listing prototypes will appear. Choose one from the pop-up menu, and the newly created object will have the prorotype applied to it.

Object Box

A quick reference list of the messages understood by a given object can be seen by Option-Control-clicking on the object in a Patcher window on Macintosh or Alt-right clicking on the object on Windows.



Getting a quick list of messages understood by an object

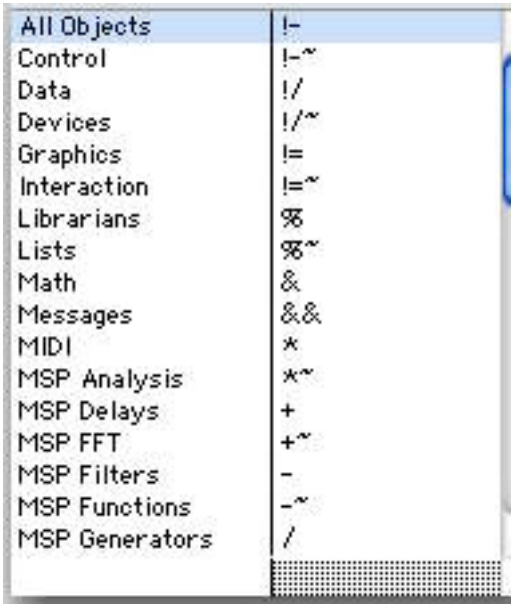
If a message requires additional arguments, the type of argument is listed after it in parentheses.

Some of the messages are shown using a terminology internal to Max. For example, in1 refers to an int in the first inlet to the right of the leftmost inlet. Furthermore, some messages shown in the quick reference message list are messages that Max sends to the object internally, *not* messages that you should send to the object in your patch. If a message appears in the quick reference message list but does not appear in the documentation, it is

probably not worth worrying about.

New Object List

Because there are so many names that may be typed into an object box, you might need some help remembering all the possible choices. When **New Object List** is checked in the Options menu, Max provides you with a complete list of all available object names each time you place a new object box in the Patcher window.



New Object List window

The New Object List window has two columns. The left column lists categories of objects, and the right section lists the names of objects in the selected category. The *All Objects* category lists all object names known to Max.

To find an object name, you can scroll through the list, type the first few letters of the desired object name, or use the up and down arrow keys on the Macintosh keyboard.

To choose an object name from the list and place it in the object box, double-click on a name, or press Return, Enter, or Space to choose the highlighted name. When an object name has been chosen, the New Object List disappears. The blinking insertion point remains inside the object box, allowing you to type in any arguments the object may require. Finally, click anywhere outside of the object box, or press the Enter key on Macintosh or the Shift and Enter keys on Windows, and the new object will be created.

When you are in the New Object List window, you can get help on the selected object name by holding down the Option key on Macintosh or the Alt key on Windows while you double-

click, press Return, Enter, or Space.

If you want the New Object List to disappear to type in an object name yourself, press the Delete (Backspace) key, or click anywhere outside the New Object List. The blinking insertion point will be placed back inside the object box, so that you can type in the name. To create a new object without even seeing the New Object List, hold down the Option key on Macintosh or the Alt key on Windows as you place the object box in the Patcher window, or uncheck **New Object List** in the Options menu.

Editing the Max New Object List

The *Max New Object List* shows a complete list of available objects whenever you create a new object box. You can customize the names in the New Object List, adding your own names that you might type often. The objects you add can be external Max objects, or they can be your own patches you have created and saved as Max documents. You can even create your own categories of objects or patches.

To add your objects to Max's New Object List, you edit the specially-formatted text files in the init folder located in the /Library/Application Support/Cycling '74 folder on Macintosh or in the C:\Program Files\Common Files\Cycling '74 folder on Windows. The file contains a list of messages to max that are loaded when the application is launched. If you'd like to see the standard categories that the New Object List already uses, you can look at the files "max-objectlist.txt" (which contains the standard Max objects) or "audio-objectlist.txt" (which contains the standard MSP objects).

You can also create your own text file that contains your own new categories and object lists. You can have as many text files in the init folder as you need.

Entries in a text file loaded into the New Object List look like this:

```
max oblist <category> <objectname>;
```

If the name of a category (or object) includes spaces or other special characters, the name should be contained within double quotes:

```
max oblist "My Special Objects" mixmaster;
```

If the object you want to add to your New Object List is available for only Windows or Macintosh, You can add system windows or system macintosh after the word "max." That way, you can use the same object list files on both platforms, and you'll only see the objects available for the platform you're running Max on.

```
max system windows oblist "Secret Weapons" cantileverxp.ext
```

After editing the Max Object List file, you need to restart Max before it will change the appearance of the New Object List window.

There are no restrictions on the category name, but the object name will not create a valid object after being entered into an object box if it contains a comma or semicolon. You can add arguments after names of objects if you want them entered to an object box.

Adding categories can be useful to provide you with the names of objects you access frequently. To see all the categories, use the grow box at the bottom-right corner of the New Object List window to make the window tall enough to see all of your categories. The size of the window is saved as a preference for the next time you create an object box, and the expanded window size will be also used the next time you launch Max.

Two Kinds of Max Objects

In various places in this manual, you may see references to *built-in* and *external* objects. The differences are minor but important. Built-in objects are part of the Max application.

External objects are separate files that contain C code that can be "linked into" the Max environment in such a way that it is impossible to tell that the code was not part of Max in the first place. That's why we rarely mention whether any particular object is a built-in or an external in this manual.

In addition, there are two kinds of Max objects:

- *normal* objects—Max objects that appear in a Patcher window as an object box that contains its name or a character (e.g., **borax**, **&&**)

- *user interface* objects—Max objects that appear as icons in the Patcher window palette and have some distinctive graphical appearance in a Patcher.

External user interface objects should be placed in your max-startup folder so their icons are available in the palette from the time you begin working. Normal external objects can be placed in another folder in Max's search path and will be loaded as they are needed.

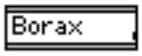
Normal external objects can provide access to non-MIDI hardware devices, and they may also implement capabilities not found in the basic set of Max objects (such as the **coll** object that allows the storage and retrieval of arbitrary list structures indexed by numbers or symbols). External user interface objects, such as the Keyboard Slider **kslider**, are important for constructing custom interfaces within Patcher windows.

Keep External Objects Where They Can Be Found

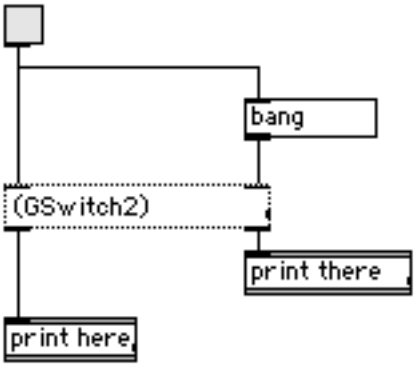
Max needs to be able to find external object files when loading a document. When an external object cannot be located, you'll see the following error in the Max window:

> • error: objectname: no such object

In the patcher you loaded, you'll see an object box with gray lines at the top and bottom (i.e., with no inlets or outlets) with the name of the object in it if it's not a user interface object.



If you load a patch that includes a user interface object that cannot be found, you'll see a gray-outline box that contains the name of the object in parentheses.



If you load a patch that contains references to missing external user interface objects, the wiring of the patch may get messed up when the patch is opened. Trying to load a patch full of missing user interface objects that also contains embedded **preset** objects could result in a large stream of unhappy error messages in the Max window.

In order for Max to find external objects automatically, they must be located in a folder specified in the File Preferences window. To add folders, choose **File Preferences...** from the Options menu.

You can manually load external objects into the Max environment by double-clicking on them or choosing **Install…** from the File menu and using the standard Open File dialog to locate the external object file you want to install.

If you are in the process of developing an external object, you cannot use **Install…** to add a "revised" copy of an external object into the environment—Max will continue to use the first object definition until you restart the application.

If you expect an external object to be loaded automatically only when it is referenced (by loading a Max document that uses it, or typing its name into an object box), the name of the external object file must be consistent with the name of the object, even though the name of the object is derived from the name of a resource inside the file.

Object Mappings

Some Max/MSP objects have names that contain characters that are reserved or have other meanings for certain OS file systems. These Max objects have file names that differ from the object names that you would type into an object box—for example, the external filename of the **!/** object is actually *rdiv*. When you type **!/** into an object box, Max uses a text file called *max-objectmappings.txt* that is loaded when you start Max to associate what you've typed with the name of the external object. The *max-objectmappings.txt* contains a complete listing of file mappings. This and other object-mapping files can be found in /Library/Application Support/Cycling '74/init on Macintosh and C:\Program Files\Common Files\Cycling '74\init\ on Windows.

External Objects and Collectives

When you choose the menu item **Save As Collective…** from the File menu to create a collective file, the file you produce will contain the patch and all subpatcher files and external objects it uses.

In certain cases, not all external objects will be included and you may see "no such object" errors when you load your collective. This means that the standard search path containing your externals hasn't been specified—for example, a patch that uses other patcher files that are loaded dynamically with the **pcontrol** object will need to have those files included explicitly in your collective script using the patcher keyword. For more information on making collectives, see the Collectives topic in the Max Tutorials and Topics manual.

Errors When Loading External Objects

When there is an error loading an external object, Max will act the same way as if the object could not be found. The name of the object will appear in an object box with gray lines at the top and bottom (for a normal object), or within a gray rectangle with the name in parentheses (for a user interface object).

Some external objects may require the presence of additional software libraries. For instance, MSP external objects require a library called MaxAudioLib (on Macintosh, this library is built into the Max/MSP and Max/MSP Runtime applications). If you see an error message in the Max window such as

- error: can't fragload objectname: missing LibraryName (function name)

you'll need to either obtain or enable the specified library in order to use the external object(s). Doing this isn't always a straightforward procedure-some libraries may need to be placed into the Max application folder (such as Cycling '74's Jitter library), while others might

require an installation of software into your OS (such as Apple's QuickTime libraries). When in doubt, consult the documentation for the external object(s) in question.

External Objects from Third Parties

In addition to the objects that come with Max itself, there are hundreds of external Max objects that are available from third-party developers. Some are included in commercial packages, but most are free and available for you to include in non-commercial applications. You'll find many interesting collections of objects that perform everything from random number generation to video gesture recognition. We maintain a web page devoted to listing sources for external objects at

http://www.cycling74.com/community/.

Note that some third-party externals are platform-specific and they might not be available for a given platform (If you develop externals in Java using the **mxj** object, your objects will typically be cross platform as long as they do not depend on platform-specific extensions to the Java language)..

> **Note:** Before releasing a collective or application built with Max that uses a third-party external object, you should check to make sure that you are permitted to distribute it. If there is no license or use information accompanying the object, we strongly recommend that you check with the author of the object. Max objects released under the GPL (GNU Public License) cannot be included in commercial software.

If you experience problems or crashes when using a third-party external object, you should contact the author of the object. While Cycling '74 makes an effort to assist developers in creating external objects, we do not provide support for them.

Locked Patcher Window

- If **Help from Locked Patchers** in the Options menu is checked, Option-clicking on Macintosh or Alt-clicking on Windows on any object's box opens a help file for that object.

- Command-clicking on Macintosh or Control-clicking on Windows in any white space unlocks the Patcher window (if it's editable).

- Option-clicking on Macintosh or Alt-clicking on Windows on a window's close box closes all windows except the Max window.

- Option-clicking on Macintosh or Alt-clicking on Windows on the title bar of a subpatch window pops up a menu that allows you to bring any parent windows of the subpatcher to the front. If the subpatcher is an "edit-only" Patcher window (produced by double-clicking on a Patcher object which was read from a Max document), the top item in the list opens the Max document for editing.

- Typing Command-period on Macintosh or Control-period on Windows stops the Max scheduler, allowing you to recover from a runaway process which might be taking up too much CPU time to stop by normal methods. After you have remedied the situation which caused the process to get out of hand, choose **Resume** from the Edit menu to restart the scheduler.

- Holding down both the Command and Shift keys on Macintosh or the Control and Shift keys on Windows while a patch is loading prevents **loadbang** objects in that patch from sending any output.

Unlocked Patcher Window

Contextual Menus

- Control-clicking on Macintosh or Right-clicking on Windows in a Patcher window brings up a menu of useful editing commands. For more information about the Patcher window contextual menus, see the Menus topic.

- Option-Control-clicking on Macintosh or Alt-Right-clicking on Windows on any object displays a menu of all the messages you can send to an object. For more information on this menu, see the Menus topic.

Shortcuts

Selecting and Moving Objects

- Option-clicking on Macintosh or Alt-clicking on Windows on any object's box opens a help file for that object.
- Shift-clicking on an object box reverses the selected state of the object without changing the selected state of other objects.
- Shift-dragging an object box helps constrain the dragging of the object in the horizontal or vertical dimension.
- Option-clicking on Macintosh or Alt-clicking on Windows on one or more objects and dragging will duplicate the object(s).
- Hold down the shift key when clicking to place an object. Except for the object box, the cursor will remain the same, so you can make multiple copies of the same object.
- The arrow keys move the selected boxes by 1 pixel in any direction.
- Option-dragging a rectangle around a set of objects and patch cords or Alt-dragging on Windows selects both the patch cords and the objects.
- To drag a text object which has been selected for editing, move the cursor to the top or bottom edge of the box, then drag, as shown below. This is a handy way to move an object after duplicating it.

Click at the top or bottom edge of a selected text box to drag it

- Command-clicking on Macintosh or Control-clicking on Windows on any user interface object, such as a **slider** or **number box**, operates the object as if the Patcher window were locked.
- Command double-clicking on Macintosh or Control double-clicking on Windows edits objects such as **patcher**, **table**, and **coll** that open when you double-click them in a locked Patcher window.
- Command-clicking on Macintosh or Control-clicking on Windows in any white space locks or unlocks the Patcher window.

- With no objects selected, holding down the Option key on Macintosh or the Alt key on Windows and choosing **Send to Back** from the Object menu sends all **comment** objects to the background so they will not appear in front of other objects when the Patcher is locked. This may be helpful for updating files from previous versions of Max in which **comment** objects appear in front of other objects.

- After typing into an object or message box, Option-clicking on Macintosh or Alt-clicking on Windows outside the box prevents Auto Fix Width from changing the box's size. Option- clicking on Macintosh or Alt-clicking on Windows outside a comment box invokes Auto Fix Width on the comment, where it is normally disabled.

Patch Cord Shortcuts

- If Segmented Patch Cords is not checked in the Options menu, shift-clicking on an outlet of an object uses **Segmented Patch Cords** mode, in which subsequent clicks define the "corners" of the patch cord.

- Shift-clicking on an inlet of an object when making a connection lets you make multiple connections from a single outlet; another patch cord will be created immediately for the next connection.

- If **Segmented Patch Cords** is checked in the Options menu, shift-clicking uses the normal mode of dragging from outlet to inlet to make a straight patch cord.

- Hold down the control key if you are making a segmented patch cord and want to make a corner over an object—the normal auto-connection feature will be disabled.

- Command-clicking on Macintosh or Control-clicking on Windows while making a segmented patch cord gets rid of the patch cord and cancels the operation.

- Option-clicking on Macintosh or Alt-clicking on Windows while making a segmented patch cord gets rid of the last segment.

Creating Objects

- Option-clicking on Macintosh or Alt-clicking on Windows after selecting the Object Box tool in the palette places the object box without showing the New Object List window.

- Pressing the Delete (Backspace) key after selecting a tool from the palette cancels the operation and returns to the normal cursor.

- Clicking on the white area at the far left of the palette also cancels the selected palette tool.

Shortcuts

New Object List

- Option-clicking on Macintosh or Alt-clicking on Windows on an empty object box opens the New Object List window.
- Delete (Backspace) hides the New Object List window.
- The Space bar enters the text of the selected item in the New Object List into the object box and adds a space afterwards for typing in any arguments.
- Return or Enter enters the text of the selected item in the New Object List into the object box.
- The Up and Down Arrow Keys scroll the selected item up and down.
- Tab switches which column of items is affected by typing.
- Holding the Option key on Macintosh or the Alt key on Windows down while double-clicking or typing Return, Space, or Enter opens a help file on the selected item.

send, receive, and value

- Double-clicking on a **send**, **receive**, or **value** object provides a contextual menu with a list of instances of these objects.

Table Editing Window

- Command-clicking on Macintosh or Control-clicking on Windows with the *Pencil* tool magnifies the area around where you clicked. If you are at 8:1 x 8:1 zoom, Command-clicking on Macintosh or Control-clicking on Windows returns to 1:1 x 1:1 magnification, or the minimum allowed zoom above 1:1.

Any Window

- Command-clicking on Macintosh or Control-clicking on Windows in any window while **All Windows Active** is enabled brings that window to the front (if the window is not already in front).

  Option-clicking on Macintosh or Alt-clicking on Windows on the close box of a window closes all windows except the Max window.

Shortcuts

Inspectors

- The Cut, Copy, and Paste keyboard shortcuts using the Command keys on Macintosh or the Control keys on Windows work in the text fields of any object's Inspector window.

MIDI Messages

A MIDI message is composed of a *status* byte, which identifies the message type, followed in most cases by one or more *data bytes*. The most significant bit of data bytes is always 0, to distinguish them from status bytes, whose most significant bit is 1. Except for System Exclusive messages, the status byte specifies exactly how many data bytes will follow it. System Exclusive messages are used for synthesizer patch dumps and parameter changes.

MIDI Objects

Max MIDI objects extract the essential data from incoming messages, so you don't need to know the details of the structure of MIDI messages. These objects are listed in the table below describing each type of MIDI message.

Raw MIDI

If you wish to deal with receiving and transmitting entire MIDI messages yourself, you can use the **midiin** and **midiout** objects. The **midiparse** and **midiformat** objects filter and format raw MIDI, and **sxformat** can help in formatting system exclusive commands for transmission by **midiout**.

To help you in managing MIDI data in Max, we have provided two reference charts here. The first chart identifies the different types of MIDI messages and shows their format. The second chart identifies controller numbers (the second byte of a MIDI control change message) that have been assigned a specific function.

MIDI Overview

MIDI Messages

Channel Messages Channel Messages use the lower 4 bits of the status byte to indicate the MIDI channel of the message. 0 is MIDI channel 1, and 15 is MIDI channel 16						
Function	*Objects*	*Status Byte*			*2nd Byte (0-127)*	*3rd Byte (0-127)*
		Decimal	*Hex*	*Binary*		
Note Off	**xnotein, xnoteout**	128-143	80-8F	1000xxxx	Key Number	Release Velocity
Note On	**notein, noteout**	144-159	90-9F	1001xxxx	Key Number	Velocity
Poly Pressure	**polyin, polyout**	160-175	A0-AF	1010xxxx	Key Number	Aftertouch
Control Change	**ctlin, ctlout**	176-191	B0-BF	1011xxxx	Controller Number	Controller Data
Program Change	**pgmin, pgmout**	192-207	C0-CF	1100xxxx	Program Number	
Aftertouch	**touchin, touchout**	208-223	D0-DF	1101xxxx	Aftertouch Value	
Pitch Bend	**bendin, bendout**	224-239	E0-EF	1110xxxx	Bend (LSB)	Bend (MSB)

System Messages						
Function	*Objects*	*Status Byte*			*2nd Byte (0-127)*	*3rd Byte (0-127)*
		Decimal	*Hex*	*Binary*		
System Exclusive	**sysexin, midiout**	240	F0	11110000	Mfr. ID Number	Arbitrary
Song Pos Ptr	**midiin, midiout**	242	F2	11110010	Position (LSB)	Position (MSB)
Song Select	**midiin, midiout**	243	F3	11110011	Song Number	
Tune Request	**midiin, midiout**	246	F6	11110110		
End of Sys Ex	**sysexin, midiout**	247	F7	11110111		
Clock	**rtin, midiout**	248	F8	11111000		
Start	**rtin, midiout**	250	FA	11111010		
Continue	**rtin, midiout**	251	FB	11111011		
Stop	**rtin, midiout**	252	FC	11111100		
Active Sensing	**midiin, midiout**	254	FE	11111110		
System Reset	**midiin, midiout**	255	FF	11111111		

MIDI Overview

Control Changes

Function	*Ctl Num*	*Values*
Continuous Controllers (MSB)	**0-31**	**0-127**
Modulation Wheel	1	0-127
Breath Controller	2	0-127
Foot Controller	4	0-127
Portamento Time	5	0-127
Data Entry	6	0-127
Main Volume	7	0-127
Balance	8	0-127
Pan	10	0-127
Expression	11	
Extra precision for the above (LSB)	**32-63**	
On/Off Switch Controllers	**64-95**	**127 and 0**
Sustain Pedal	64	127 and 0
Portamento On/Off	65	127 and 0
Sostenuto Pedal	66	127 and 0
Soft Pedal	67	
Other	**96-121**	**127**
Data Entry Yes (+1)	96	127
Data Entry No (-1)	97	
Channel Mode Messages	**122-127**	**127 and 0**
Local Control On/Off	122	0
All Notes Off	123	0
Omni Mode Off	124	0
Omni Mode On	125	0-16
Mono On	126	0
Poly On	127	

Using MIDI

On Windows, all MIDI devices which are installed correctly on your system and appear in the Sounds and Audio Devices Properties (Start - Settings - Sounds and Audio Devices) will be available to Max/MSP for MIDI I/O.

On Macintosh, the OS X application Audio MIDI Setup, located in the Utilities folder inside the Applications folder, is used to set up and describe your MIDI setup.

In addition to the standard MIDI drivers used to communicate with external MIDI gear (coremidi on the Mac and midi_mme on Windows), there are additional MIDI drivers available for use:

midi_adrewire — This driver sends and receives MIDI from a ReWire host application when Max is a ReWire client

augraph — (Macintosh) This driver addresses DLS synthesizers built into the Macintosh platform's operating system.

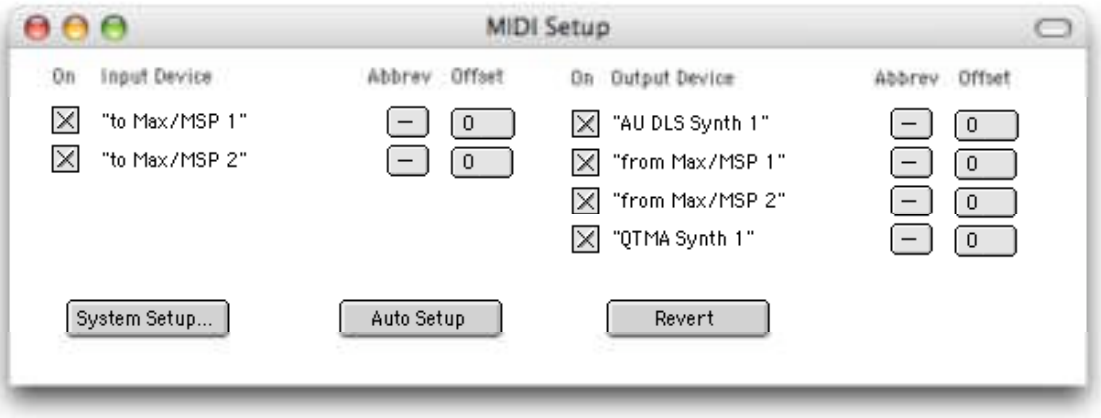
midi_dm — (Windows), This driver addresses DLS synthesizers built into the Macintosh platform's operating system.

Max allows you to have multiple active MIDI driver objects in operation.

Using MIDI

Using the MIDI Setup Window

The Max application specifies multiple MIDI ports by letter (a-z), or by using MIDI channel numbers above 16. This requires a certain amount of translation between Core MIDI device names and the Max lettering scheme. You can use the MIDI Setup window to perform this translation. To see the MIDI Setup window, choose **MIDI Setup...** from the File menu.



The MIDI Setup window lists the MIDI device names and lets you associate letter abbreviations and ranges of MIDI channels with them.

The default MIDI output device under Max is the internal synthesizer supported by your operating system. On the Macintosh, this is an AudioUnit DLS synthesizer that supports both its own set of internal sounds (as a General MIDI bank), as well as Level 2 SoundFont files.On Windows, this is the Microsoft DirectMusic DLS synthesizer. Note that the Microsoft DLS synthesizer does not support SoundFont files. For more information about working with DLS synths, see the section **Using the DLS Synthesizer** later in this chapter.

To assign an abbreviation and MIDI channel range to a device, select the desired values from the pop-up menus. The abbreviation allows you to type the letter instead of the device name as an argument to MIDI objects such as **notein** or **noteout**. The channel offset is added to the MIDI channel of the incoming data to distinguish its device for MIDI objects set to receive from multiple devices or sent to multiple devices. For example, if a device is given a channel offset of 32, a note-on message sent to Max will be output on channel 33 of a **notein** object. Similarly, if you send 33 to the MIDI channel inlet of a **noteout** object, a subsequent int sent to the left inlet will produce a note message on MIDI channel 1 for the device whose channel offset is 32.

Using MIDI

Default Devices for MIDI Objects

If you create a MIDI output object without specifying a device name, the object transmits MIDI to the first device in the list of output devices in the MIDI Setup window.

A MIDI input object that is not assigned to a specific port (i.e., one that does not have a MIDI device name or abbreviation as an argument) will merge all input devices. In that case, the only way for a Max patch to determine which device is actually the source of input to these objects is to compare the incoming MIDI channel number to the MIDI channel offset specified for each device. Since you may wish to treat all MIDI input identically regardless if its source, this may be useful.

The **midiin** object is an exception to this behavior—it receives data only from the first device in the input device list if no device is specified in an argument. If multiple devices share the same letter abbreviation, Max will use the first one in the list at the time a MIDI input or output object is created with that abbreviation as an argument. Changing the abbreviations of devices has no effect on pre-existing objects, although it will have an effect on the meaning of subsequent port messages sent to MIDI output objects with abbreviations as argument.

Easy Default Setup

To set all devices' abbreviations and channel offsets, click the *Auto Setup* button. Max will create a standard set of abbreviations and channel offsets for both input and output devices. The first device in each list will be given an abbreviation of a and a channel offset of 0, the second device will be given an abbreviation of b and a channel of 16, and so on.

Using the DLS Synthesizer

Both the Macintosh and Windows XP provide a DLS (Downloadable Soundfont) synthesizer for MIDI playback. If you don't want to use any external MIDI gear, you can drive the DLS synth directly from Max via MIDI.

By default, a single *augraph* (on Mac OS X) or *midi_dm* (on Windows) port is created. However, you can create additional MIDI synthesizer ports and assign new DLS sound bank files to each one. Addressing the DLS synthesizers currently requires the use of the **message** box technique where you send messages to named objects by typing a semicolon followed by the message text into a message box, then click the message box.

Here is a list of messages that you can use to access the DLS synthesizer:

Creating a Port:

```
;#SM createoutport <portname> <drivername>
```

where *drivername* is midi_dm on Windows and augraph on the Macintosh. *portname* is the name you assign to the port. For example:

```
; #SM createoutport myOtherSynth midi_dm

; #SM createoutport myOtherSynth augraph
```

Deleting a Port:

```
;#SM deleteoutport <portname> <drivername>
```

where *drivername* is midi_dm on Windows and augraph on the Macintosh. *portname* is the name of your choice. For example:

```
; #SM deleteoutport myOtherSynth midi_dm

; #SM createoutport myOtherSynth augraph
```

Loading a DLS Bank (type 1 or 2)

```
;#SM driver loadbank <filename> <portname>
```

where *filename* is the name of an existing DLS bank file, and *portname* is the name of the port that will use this bank. If portname is omitted, all DLS ports will use the bank. On Mac OS X, the folder /Library/Audio/Sounds/Banks is added to the search path when looking for a DLS bank file.

Loading the Default GM Bank:

```
;#SM driver loadbank 0 <portname>
```

Turning Reverb On and Off

```
;#SM driver reverb 1/0 <portname>
```

By default reverb is off in both augraph and midi_dm.

Setting MIDI Output Latency (midi_dm only)

```
;#SM driver latency <time> <portname>
```

where *time* is a value in milliseconds and *portname* is the port that is set to this value. For example, the following message would set the latency to 10 milliseconds:

```
;#SM driver latency 10 portname
```

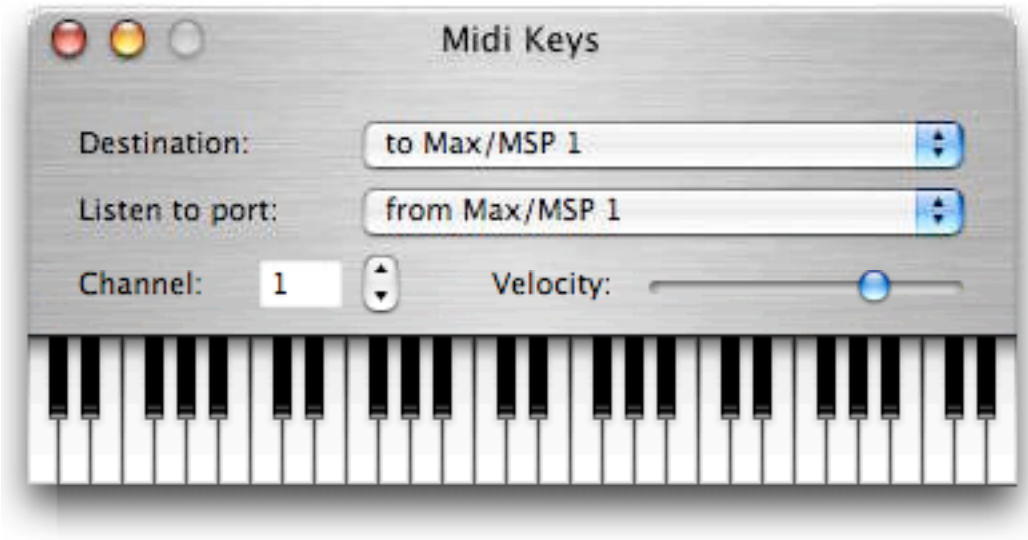
Virtual Input and Output Devices on Macintosh

Max has two input and output devices you can use to send MIDI to other programs running your computer, or to send MIDI to Max from other programs. The output devices are *virtual sources.*

They are labeled on the Macintosh MIDI setup window as "from Max/MSP 1" and "from Max/ MSP 2". They are sources of MIDI for other programs, and can be selected as MIDI inputs in another application to establish a MIDI connection from Max.

The input devices labeled "to Max/MSP 1" and "to Max/MSP 2" are known as *virtual destinations*, because you can choose them as MIDI output devices in other applications.

Here is an example that uses the Macintosh MIDIKeys application to send MIDI to Max. The Destination is listed as to Max/MSP 1, so when you click on the keyboard, MIDI will be sent to Max's virtual destination.



Similarly, the Listen to port: pop-up menu shows the Max virtual source from Max/MSP 1. If

MIDI notes were sent from Max to the from Max/MSP 1 device, they would be displayed on the Midi Keys keyboard.

Adding Virtual MIDI Ports

By default, Max creates two "virtual" MIDI ports for both input and output. If you wish to add additional virtual ports to your MIDI system, you can use the same **message** box technique described above to send the createoutport and createinport messages.

```
;#SM createoutport <portname> <drivername>

;#SM createinport <portname> <drivername>
```

where *portname* is the name you assign to the port, and *drivername* is the driver to be used (currently, only CoreMIDI is supported). For example, the following two messages:

```
;#SM createoutport myvirtualport CoreMIDI

;#SM createinport myvirtualport CoreMIDI
```

Would create a virtual MIDI input and output port, each named "myvirtualport". These virtual ports are not saved as part of the Max/MSP setup, so they will have to be recreated each time you restart Max.

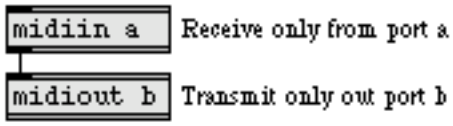
See Also

Ports How MIDI ports are specified

setclock Control the clock speed of timing objects remotely

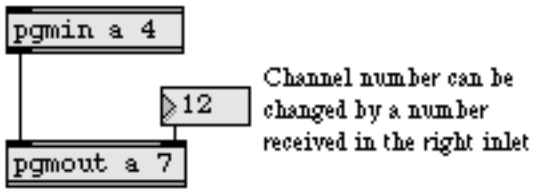
Max's MIDI Objects

Each object in Max that transmits or receives MIDI data can be set to communicate with a specific port by typing in a letter argument after the name of the object.



Any letter from a to z can be used, provided that letter has been assigned to a device (or a virtual port) in the MIDI Setup dialog box. Refer to the MIDI chapter for information about assigning abbreviations and MIDI channel ranges.

For MIDI objects that receive and transmit specific channel voice messages, such as **notein** and **bendout**, a letter argument followed by a number argument indicates a port and a specific MIDI channel on which to receive or transmit MIDI messages. For transmitting objects (except **midiout**), channel numbers are received in the rightmost inlet to change the channel.



A number alone can be used in place of a letter and number combination for such objects. Numbers 1-16 specify a channel number on port a, numbers 17-32 specify channels 1-16 on port b, etc. Numbers greater than 16 can be received in the right inlet, to specify both port and channel. When a letter argument is present, however, the port is fixed, and channel numbers greater than 16 received in the right inlet are *wrapped around* to stay within the 1-16 range.

Letter argument transmits to only one port. Otherwise, number specifies both port and channel.

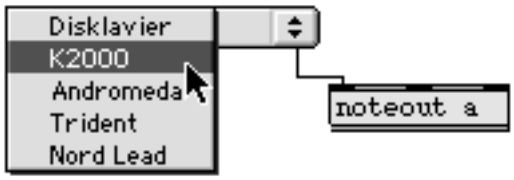
The ports can also be assigned an arbitrary range of sixteen MIDI channels. So, for example, port b could be assigned channels 1-16 and port a could be assigned channels 17-32.

Specifying MIDI Ports

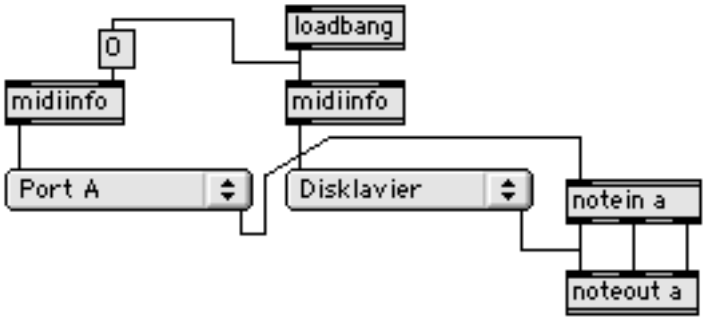
MIDI ports may be specified three ways: by device name, letter abbreviation (a to z), or MIDI channel number. Device names are used as arguments to MIDI objects in place of letters. If your device names contain spaces, you will need to include the name in double quotes.

The port Message

All MIDI objects accept a message that changes the port they use to transmit or receive MIDI data. The port message takes one argument, which is the letter or device name associated with the port. The word port can be omitted, which is convenient for making a pop-up menu with port names or abbreviations, as shown below.



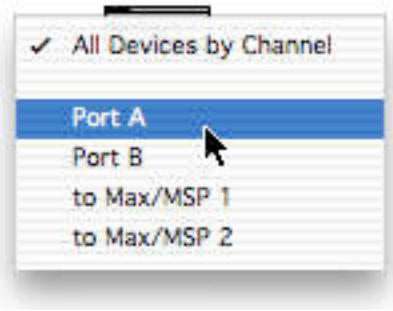
The pop-up menu can be created automatically by using the **midiinfo** object.



After receiving the port message, the MIDI output object acts as if the port had been specified as an argument—all MIDI channels above 16 wrap around to 1-16, and data is always sent on the specified port.

Specifying Ports

Yet another way to set the port of a MIDI object is to double-click on the object when the Patcher window is locked—you'll be presented with a menu containing available MIDI ports. You also have the option to choose *All Devices by Channel* if you want to be able to identify or specify different devices by means of channel number alone.



See Also

Using MIDI Using Max with MIDI

Index

Index

Index

Index