

Max API

8.2.0

Cycling '74

# Chapter 1

## Objects in C: A Roadmap

Max has an extensive API for developing new objects in C. Before you start learning about it, however, we would like to save you time and make sure you learn the minimum about the API for what you need to do. Therefore, we've made a brief list of application areas for object development along with the sections of this document with which you'll probably want to become familiar.

### 1.1 Max Objects

For **logic and arithmetic objects**, such as new mathematical functions or more complex conditional operations than what is offered in Max, it should be sufficient to read the [Anatomy of a Max Object](#) section.

For objects that use [Data Structures](#), you'll want to read, in addition, the [Atoms and Messages](#) section to learn about Max's basic mechanisms for representing and communicating data.

If you are interested in writing interfaces to **operating system services**, you may need to learn about Max's [Threading](#) model and [The Scheduler](#).

For objects that deal with time and timing you'll want to learn about [The Scheduler](#). If you're interested in tempo-based scheduling, you'll want to read the section on [ITM](#) and look at the `delay2` example.

To create new user interface gadgets, you'll want to read all of the above, plus the section on [Attributes](#) and the [Anatomy of a UI Object](#). The section on [JGraphics](#) will also be helpful.

To create **objects with editing windows**, things are much more complicated than they used to be. You'll need to learn everything about UI objects, plus understand the `scripto` example object project.

For patcher scripting and interrogation objects, the section on [Scripting the Patcher](#), plus a few of the examples will be very helpful. It is also helpful to have a clear conceptual understanding of the patcher, which might be aided by reading the patcher scripting sections of the `js` object documentation.

Max 6 introduced support for passing structured data with the [Dictionary Passing API](#).

## 1.2 MSP Objects

To create audio **filters** and **signal generators**, read the [Anatomy of a Max Object](#), then read the [Anatomy of a MSP Object](#) section. MSP objects make use of [Creating and Using Proxies](#) when receiving multiple audio inputs, so familiarity with that concept could be helpful.

For audio objects that output events (messages), you'll need to use the services of [The Scheduler](#), so we suggest reading about that.

For UI objects for analyzing and controlling audio, you'll need to learn about regular MSP objects as well as Max UI objects.

Information on updating MSP objects from Max 5 or earlier for 64-bit audio (introduced in Max 6) is located in [Appendix: Updating Externals for Max 6](#).

Max 8 introduced MC for working with multi-voice/multi-channel signals. More information is available in the [MC](#) chapter.

## 1.3 Jitter Objects

The [Jitter Object Model](#) outlines some important basic information about Jitter's flexible object model. [Jitter Max Wrappers](#) describes how to write Max wrapper objects that contain Jitter objects for use in the Max patcher world. [Matrix Operator QuickStart](#) and [Matrix Operator Details](#) describe how to create a particular type of Jitter object called matrix operators, or MOPs. [OB3D QuickStart](#) and [OB3D Details](#) describe how to create OB3D Jitter objects for use in rendering OpenGL scenes. [Scheduler and Low Priority Queue Issues](#) covers important threading and timing issues when building Jitter objects. [Jitter Object Registration and Notification](#) explains how Jitter objects can be registered by name and notify clients as they change or important events occur. [Using Jitter Objects in C](#) provides some examples of how to instantiate and take advantage of Jitter objects from C, just as one would from Java, Javascript, or the patcher. Finally, The [JXF File Specification](#) and [Jitter Networking Specification](#) contain information relating to the data formats involved in the JXF file format and Jitter networking protocols, respectively.

## Chapter 2

# Development System Information

### 2.1 Building

This SDK documentation is accompanied by a series of projects for compiling some example Max external objects. The details of how to build these projects are documented below in separate sections for the [Mac](#) and [Windows](#).

When you build the example projects, the resulting Max external will be located in a folder called "externals" at the top level of the SDK package. This is located a couple of folders up from the project and source file.

Place the SDK package in your "Packages" folder in order to load and test the externals you build in Max itself.

### 2.2 Build Script

We have provided a basic script that will build all of the projects in the SDK at once. This script is written using the Ruby language. A Ruby interpreter is standard on the Mac. On windows you can download and install Ruby (1.9.3 recommended) from <http://rubyinstaller.org/> .

Run from Terminal.app (Mac) or a Command Prompt (Windows) by cd'ing into the examples directory, and then running:

```
ruby build.rb
```

### 2.3 Mac

Max external objects for the Mac are Mach-O bundles (folders that appear to be files) whose filenames must end with the .mzo extension. The example projects have been tested using Xcode 6.2. Xcode is available through the Apple Mac Store.

After installing Xcode, if you wish to run the aforementioned Ruby script, you will also need to install the Command Line Tools. This is done via the menu item: Xcode > Open Developer Tool > More Development Tools...

### 2.3.1 XCode Project Setup

The example projects are set up to have Development and Deployment build configurations. The Development configuration does not optimize and ensures debugging symbols are present. The Deployment configuration creates a universal binary and performs optimization.

The files required for the projects are included in the project folders with the exception of the following two files:

- Info.plist
- maxmspsdk.xcconfig

These two files are located one folder-level up from the project folder, and are required for the Xcode project to build the Max external.

### 2.3.2 Linking and Frameworks

External objects use dynamic linking to access the API functions provided by the Max application. When an object is loaded, calls to functions inside the application are resolved by the operating system to the correct memory address. Due to the fact that "Max" could exist as an application, a standalone you create, or a library inside another application, an object's Xcode project does not link directly to a framework, library, or application. Instead, the list of permitted symbols is provided to the linker. This list is defined in the aforementioned maxmspsdk.xcconfig file.

Audio objects will link against MaxAudioAPI.framework and Jitter objects link against JitterAPI.framework. Alternatively, you could also provide linker flags as we have provided for Max itself. The most recent version of all frameworks will be found inside the application you are using (they are found inside the application bundle in Contents/Frameworks). In addition, there are versions inside the c74support folder provided with the SDK. These will be used only to link your objects; they are never actually executed.

Xcode uses something called the Frameworks Search Path to locate frameworks when linking. The example SDK projects use a frameworks search path with a c74support folder two levels up from your the folder containing your Xcode project. If you rearrange the SDK folders, projects may not find the frameworks and will fail to link properly. Furthermore, even though we specify the frameworks search path, Xcode seems to look in /Library/Frameworks first. If you have installed a version of the Max SDK for version 4.6 or earlier, you may have older versions of MaxAPI.framework and MaxAudioAPI.framework in /Library/Frameworks. When you try to link objects that contain references to functions only defined in the newest MaxAPI.framework, the link may fail because the projects are using the old frameworks. To fix this, you'll need to remove the Max frameworks from /Library/Frameworks. If you want to develop objects for both the Max 4.6 and Max 5 SDKs on the same machine, you'll need to modify your 4.6 projects to specify a Frameworks Search Path, and relocate the 4.6 frameworks to the specified location.

## 2.4 Windows

Max external objects for Windows are Dynamic Link Libraries (DLLs) whose filenames must end with the .mxe extension (for 32-bit builds) or .mxe64 (for 64-bit builds). These DLLs will export a single function called "ext\_main" which is called by max when the external object is first loaded. Generally these DLLs will import functions of the Max API from the import library "MaxAPI.lib" which is located in the c74support\max-includes\ folder. External objects that use audio functionality will import functions from the import library "MaxAudio.lib" which is located in c74support\msp-includes\ External objects that use Jitter functionality will import functions from the import library "jitlib.lib" which is located in c74support\jit-includes\.

### 2.4.1 Compiling with Visual Studio

The example projects are in Visual C++ 2013 format (vcxproj files). A free version of Visual C++ can be obtained from Microsoft at <http://www.microsoft.com/express/>. You will want to choose "Visual Studio Express 2013 for Windows Desktop".

The projects are set up to have both a Debug and a Release configuration. The Release configuration is optimized whereas the Debug one is not. Note that for debugging purposes you can exercise your object in the Max Runtime since the copy protection for the Max Application will interfere when run under the debugger.

Another thing to note is that Max has a private build of the Microsoft C Runtime Library for historical and backward compatibility reasons. It is recommended that you link with Microsoft's standard C runtime library rather than the Max C runtime library. When you include "ext.h" from the max API it will include ext\_prefix.h which for the release build will automatically cause your project to use the max C runtime library. To use the Microsoft C Runtime define the C preprocessor macro MAXAPI\_USE\_MSCRT before including ext.h.

## 2.5 Important Project Settings

The easiest way to create a new external is to choose one of the existing SDK examples, duplicate it, and then change only the settings that need to be changes (such as the name of the project). This will help to guarantee that important project settings are correct. Project settings of particular importance are noted below.

### 2.5.1 Mac

Particularly important for Max externals on the Mac are that the Info.plist is correct set up and that the "Force Package Info Generation" is set to true. Without these your object may fail to load on some machines.

### 2.5.2 Windows

In the preprocessor definitions for the Visual Studio project it is important to define WIN\_VERSION and EXT\_WIN\_←VERSION to ensure that the headers are set up properly.

## 2.6 Platform-specificity

If you are writing a cross-platform object and you need to do something that is specific to one platform, the Max API headers provide some predefined symbols you can use.

```
#ifdef MAC_VERSION
// do something specific to the Mac
#endif
#ifndef WIN_VERSION
// do something specific to Windows
#endif
```

Another reason for conditional compilation is to handle endianness on the Mac platform. If you are still supporting PowerPC, you may have situations where the ordering of bytes within a 16- or 32-bit word is important. ext\_byteorder.h provides cross-platform tools for manipulating memory in an endian-independent way.

## 2.7 Configuration

As the Max API evolves, the use of a number of older legacy functions are discouraged. Use of said functions will issue a 'deprecation' warning when you try to compile the code. To disable these deprecation warnings you can define the preprocessor symbol C74\_NO\_DEPRECATED in the target preprocessor section of your IDE.



## Chapter 3

# Anatomy of a Max Object

Max objects are written in the C language, and the Max API is C-based.

You could use C++ but we don't support it at the API level. Writing a Max object in C, you have five basic tasks:

- 1) including the right header files (usually ext.h and ext\_obex.h)
- 2) declaring a C structure for your object
- 3) writing an initialization routine called ext\_main that defines the class
- 4) writing a new instance routine that creates a new instance of the class, when someone makes one or types its name into an object box
- 5) writing methods (or message handlers) that implement the behavior of the object

Let's look at each of these in more detail. It's useful to open the simplemax example project as we will be citing examples from it.

### 3.1 Include Files

Most of the basic Max API is included in the files ext.h and ext\_obex.h. These are essentially required for any object. Beyond this there are specific include files for more specialized objects.

The header files are cross-platform.

- jpatcher\_api.h is required for any Max UI objects
- z\_dsp.h is required for MSP audio objects

```
#include "ext.h" // should always be first, followed by ext_obex.h and any other files.
```

## 3.2 The Object Declaration

Basic Max objects are declared as C structures. The first element of the structure is a [t\\_object](#), followed by whatever you want. The example below has one long structure member.

```
typedef struct _simp
{
    t_object s_obj;      // t_object header
    long s_value;        // something else
} t_simp;
```

Your structure declaration will be used in the prototypes to functions you declare, so you'll need to place above these prototypes.

## 3.3 Initialization Routine

The initialization routine, which must be called `ext_main`, is called when Max loads your object for the first time. In the initialization routine, you define one or more classes. Defining a class consists of the following:

- 1) telling Max about the size of your object's structure and how to create and destroy an instance
- 2) defining methods that implement the object's behavior
- 3) in some cases, defining attributes that describe the object's data
- 4) registering the class in a name space

Here is the `simp` class example initialization routine:

```
static t_class *s_simp_class; // global pointer to our class definition that is setup in ext_main()
void ext_main(void *r)
{
    t_class *c;
    c = class_new("simp", (method)simp_new, (method)NULL, sizeof(t_simp), 0L, 0);
    class_addmethod(c, (method)simp_int, "int", A_LONG, 0);
    class_addmethod(c, (method)simp_bang, "bang", 0);
    class_register(CLASS_BOX, c);
    s_simp_class = c;
}
```

In order for Max to call the `ext_main()` function on your compiled external, that function must be "exported" or made public. This is accomplished by using the `C74_EXPORT` macro in the prototype of the `ext_main()` function, which is provided for you automatically in the "ext.h" header file.

`class_new()` creates a class with the new instance routine (see below), a free function (in this case there isn't one, so we pass NULL), the size of the structure, a no-longer used argument, and then a description of the arguments you type when creating an instance (in this case, there are no arguments, so we pass 0).

`class_addmethod()` binds a C function to a text symbol. The two methods defined here are `int` and `bang`.

`class_register()` adds this class to the `CLASS_BOX` name space, meaning that it will be searched when a user tries to type it into a box.

Finally, we assign the class we've created to a global variable so we can use it when creating new instances.

More complex classes will declare more methods. In many cases, you'll declare methods to implement certain API features. This is particularly true for UI objects.

## 3.4 New Instance Routine

The standard new instance routine allocates the memory to create an instance of your class and then initializes this instance. It then returns a pointer to the newly created object.

Here is the simp new instance routine

```
void *simp_new()
{
    t_simp *x = (t_simp *)object_alloc(s_simp_class);
    x->s_value = 0;
    return x;
}
```

The first line uses the global variable s\_simp\_class we defined in the initialization routine to create a new instance of the class. Essentially, the instance is a block of memory of the size defined by the class, along with a pointer to the class that permits us to dispatch messages correctly.

The next line initializes our data. More complex objects will do a lot more here, such as creating inlets and outlets. By default, the object being created will appear with one inlet and no outlets.

Finally, in the last line, we return a pointer to the newly created instance.

## 3.5 Message Handlers

We are now ready to define some actual behavior for our object by writing C functions that will be called when our object is sent messages. For this simple example, we will write only two functions. simp\_int will be called when our object receives numbers. It will store the received number in the s\_value field. simp\_bang will be called when our object receives a bang. It will print the value in the Max window. So, yes, this object is pretty useless!

The C functions you write will be declared according to the arguments the message requires. All functions are passed a pointer to your object as the first argument. For a function handling the int message, a single second argument that is a long is passed. For a function handling the bang message, no additional arguments are passed.

Here is the int method:

```
void simp_int(t_simp *x, long n)
{
    x->s_value = n;
}
```

This simply copies the value of the argument to the internal storage within the instance.

Here is the bang method:

```
void simp_bang(t_simp *x)
{
    post("value is %ld", x->s_value);
}
```

The `post()` function is similar to `printf()`, but puts the text in the Max window. `post()` is very helpful for debugging, particularly when you cannot stop user interaction or real-time computation to look at something in a debugger.

You can also add a float message, which is invoked when a floating-point number is sent to your object. Add the following to your initialization routine:

```
class_addmethod(c, (method)simp_float, "float", A_FLOAT, 0);
```

Then write the method that receives the floating-point value as follows:

```
void simp_float(t_simp *x, double f)
{
    post("got a float and it is %.2f", f);
}
```



# Chapter 4

## Inlets and Outlets

You are familiar with inlets and outlets when connecting two objects together in a patcher.

To receive data in your object or send data to other objects, you need to create the C versions of inlets and outlets. In this section, we'll explain what inlets and outlets are, how to create them, and how to use them. We'll also discuss a more advanced type of inlet called a proxy that permits a message to be received in any of your object's inlets. Proxies are used by audio objects to permit inlets to handle both signals and normal Max messages.

By default, every object shows one inlet. Additional inlets appear to the right of the default inlet, with the rightmost inlet being created last.

Inlets are essentially message translators. For example, if you create an int inlet, your object will receive the "in1" message instead of the "int" message when a number arrives at this newly created inlet. You can use the different message name to define special behavior for numbers arriving at each inlet. For example, a basic arithmetic object in Max such as + stores the number to be added when it arrives in the right inlet, but performs the computation and outputs the result when a number arrives in the left inlet.

Outlets define connections between objects and are used to send messages from your object to the objects to which it is connected. What is not obvious about an outlet, however, is that when you send a number out an outlet, the outlet-sending function does not return until all computation "below" the outlet has completed. This stack-based execution model is best illustrated by observing a patch with the Max debugger window. To understand this stack-based model it may be helpful to use the breakpoint and debugging features in Max and follow the stack display as you step through the execution of a patch. Outlets, like inlets, appear in the order you create them from right-to-left. In other words, the first inlet or outlet you create will be the visually farthest to the right.

### 4.1 Creating and Using Inlets

Proper use of an inlet involves two steps: first, add a method that will respond to the message sent via the inlet in your initialization routine, and second, create the inlet in your new instance routine. (Creating inlets at any other time is not supported.)

There are three types of inlets: int, float, and custom. We'll only describe int and float inlets here because proxies are generally a better way to create an inlet that can respond to any message. For int inlets, you'll bind a function to a message "in1", "in2", "in3" etc. depending on the inlet number you assign. Here's how to create a single inlet using "in1"...

In your initialization routine:

```
class_addmethod(c, (method)myobject_in1, "in1", A_LONG, 0);
```

In your new instance routine, after calling [object\\_alloc\(\)](#) to create your instance:

```
intin(x, 1);
```

The method that will be called when an int is received in the right inlet:

```
void myobject_in1(t_myobject *x, long n)
{
    // do something with n
}
```

Creating a single inlet in this way gives your object two inlets (remember that it always has one by default). If you want to create multiple inlets, you'll need to create them in order from right to left, as shown below:

```
intin(x, 2);      // creates an inlet (the right inlet) that will send your object the "in2" message
intin(x, 1);      // creates an inlet (the middle inlet) that will send your object the "in1" message
```

Inlets that send float messages to your object are created with [floatin\(\)](#) and translate the float message into "ft1","ft2","ft3" etc. Example:

In initialization routine:

```
class_addmethod(c, (method)myobject_ft1, "ft1", A_FLOAT, 0);
```

In new instance routine:

```
floatin(x, 1);
```

Method:

```
void myobject_ft1(t_myobject *x, double f)
{
    post("float %.2f received in right inlet,f");
}
```

Note that you can mix int and float inlets, but each inlet must have a unique number. Example:

```
intin(x, 2);
floatin(x, 1);
```

## 4.2 Creating and Using Outlets

You create outlets in your new instance routine. Outlet creators return a pointer that you should store for later use when you want to send a message. As with inlets, outlets are created from right to left.

Here's a simple example. First we'll add two void pointers to our data structure to store the outlets for each instance.

```
typedef struct _myobject
{
    t_object m_ob;
    void *m_outlet1;
    void *m_outlet2;
} t_myobject;
```

Then we'll create the outlets in our new instance routine.

```
x = (t_myobject *)object_alloc(s_myobject_class);
x->m_outlet2 = bangout((t_object *)x);
x->m_outlet1 = intout((t_object *)x);
return x;
```

These outlets are type-specific, meaning that we will always send the same type of message through them. If you want to create outlets that can send any message, use [outlet\\_new\(\)](#). Type-specific outlets execute faster, because they make a direct connection to the method handler that will be called at the time you send a message. When we want to send messages out these outlets, say, in our bang method, we do the following:

```
void myobject_bang(t_myobject *x)
{
    outlet_bang(x->m_outlet2);
    outlet_int(x->m_outlet1, 74);
}
```

The bang method above sends the bang message out the m\_outlet2 outlet first, then sends the number 74 out the m\_outlet1. This is consistent with the general design in Max to send values out outlets from right to left. However, there is nothing enforcing this design, and you could reverse the statements if you felt like it.

A more general message-sending routine, [outlet\\_anything\(\)](#), will be shown in the [Atoms and Messages](#) section.

## 4.3 Creating and Using Proxies

A proxy is a small object that controls an inlet, but does not translate the message it receives. Instead it sets a location inside your object's data structure to a value you associate with the inlet. If the message comes "directly" to your object via the left inlet, the value will be 0. However, in order to be thread-safe, you should not read the value of this "inlet number" directly. Instead, you'll use the [proxy\\_getinlet\(\)](#) routine to determine the inlet that has received the message.

The advantage of proxies over regular inlets is that your object can respond to any message in all of its inlets, not just the left inlet. As a Max user, you may already appreciate the proxy feature without knowing it. For example, the pack object can combine ints, floats, lists, or symbols arriving in any of its inlets. It uses proxies to make this happen. MSP audio objects that accept signals in more than one inlet use proxies as well. In fact, the proxy capability is built into the way you create audio objects, as will be discussed in the [Anatomy of a MSP Object](#) section.

If your object's non-left inlets will only respond to ints or floats, implementing proxies is usually overkill.

## 4.4 Example

First, add a place in your object to store the proxy value. You shouldn't access this directly, but the proxy needs it. Second, you'll need to store the proxy, because you need to free it when your object goes away. If you create many proxies, you'll need to store pointers to all of them, but all proxies share the same long integer value field.

```
typedef struct _myobject
{
    t_object m_obj;
    long m_in;           // space for the inlet number used by all the proxies
    void *m_proxy;
} t_myobject;
```

In your new instance routine, create the proxy, passing your object, a non-zero code value associated with the proxy, and a pointer to your object's inlet number location.

```
x->m_proxy = proxy_new((t_object *)x, 1, &x->m_in);
```

If you want to create regular inlets for your object, you can do so. Proxies and regular inlets can be mixed, although such a design might confuse a user of your object.

Finally, here is a method that takes a different action depending on the value of x->m\_in that we check using [proxy\\_getinlet\(\)](#).

```
void myobject_bang(t_myobject *x)
{
    switch (proxy_getinlet((t_object *)x)) {
        case 0:
            post("bang received in left inlet");
            break;
        case 1:
            post("bang received in right inlet");
            break;
    }
}
```



## Chapter 5

# Atoms and Messages

When a Max object receives a message, it uses its class to look up the message selector ("int", "bang", "set" etc.) and invoke the associated C function (method).

This association is what you are creating when you use `class_addmethod()` in the initialization routine. If the lookup fails, you'll see an "object doesn't understand message" error in the Max window.

Message selectors are not character strings, but a special data structure called a symbol (`t_symbol`). A symbol holds a string and a value, but what is more important is that every symbol in Max is unique. This permits you to compare two symbols for equivalence by comparing pointers, rather than having to compare each character in two strings.

The "data" or argument part of a message, if it exists, is transmitted in the form of an array of atoms (`t_atom`). The atom is a structure that can hold integers, floats, symbols, or even pointers to other objects, identified by a tag. You'll use symbols and atoms both in sending messages and receiving them.

To illustrate the use of symbols and atoms, here is how you would send a message out an outlet. Let's say we want to send the message "green 43 crazy 8.34." This message consists of a selector "green" plus an array of three atoms.

First, we'll need to create a generic outlet with `outlet_new` in our new instance routine.

```
x->m_outlet = outlet_new((t_object *)x, NULL);
```

The second argument being NULL indicates that the outlet can be used to send any message. If the second argument had been a character string such as "int" or "set" only that specific message could be sent out the outlet. You'd be correct if you wondered whether `intout()` is actually just `outlet_new(x, "int")`.

Now that we have our generic outlet, we'll call `outlet_anything()` on it in a method. The first step, however, is to assemble our message, with a selector "green" plus an array of atoms. Assigning ints and floats to an atom is relatively simple, but to assign a symbol, we need to transform a character string into a symbol using `gensym()`. The `gensym()` function returns a pointer to a symbol that is guaranteed to be unique for the string you supply. This means the string is compared with other symbols to ensure its uniqueness. If it already exists, `gensym()` will supply a pointer to the symbol. Otherwise it will create a new one and store it in a table so it can be found the next time someone asks for it.

```
void myobject_bang(t_object *x)
{
    t_atom argv[3];
    atom_setlong(argv, 43);
    atom_setsym(argv + 1, gensym("crazy"));
    atom_setfloat(argv + 2, 8.34);
    outlet_anything(x->m_outlet, gensym("green"), 3, argv);
}
```

In the call to `outlet_anything()` above, `gensym("green")` represents the message selector. The `outlet_anything()` function will try to find a message "green" in each of the objects connected to the outlet. If `outlet_anything()` finds such a message, it will execute it, passing it the array of atoms it received.

If it cannot find a match for the symbol green, it does one more thing, which allows objects to handle messages generically. Your object can define a special method bound to the symbol "anything" that will be invoked if no other match is found for a selector. We'll discuss the anything method in a moment, but first, we need to return to `class_addmethod()` and explain the final arguments it accepts.

To access atoms, you can use the functions `atom_setlong()`, `atom_getlong()` etc. or you can access the `t_atom` structure directly. We recommend using the accessor functions, as they lead to both cleaner code and will permit your source to work without modifications when changes to the `t_atom` structure occur over time.

## 5.1 Argument Type Specifiers

In the `simp` example, you saw the int method defined as follows:

```
class_addmethod(c, (method)simp_int, "int", A_LONG, 0);
```

The `A_LONG`, 0 arguments to `class_addmethod()` specify the type of arguments expected by the C function you have written. `A_LONG` means that the C function accepts a long integer argument. The 0 terminates the argument specifier list, so for the int message, there is a single long integer argument.

The other options are `A_FLOAT` for doubles, `A_SYM` for symbols, and `A_GIMME`, which passes the raw list of atoms that were originally used to send the Max message in the first place. These argument type specifiers define what are known as "typed" methods in Max. Typed methods are those where Max checks the type of each atom in a message to ensure it is consistent with what the receiving object has said it expects for a given selector.

If the atoms cannot be coerced into the format of the argument type specifier, a bad arguments error is printed in the Max window.

There is a limit to the number of specifiers you can use, and in general, multiple `A_FLOAT` specifiers should be avoided due to the historically unpredictable nature of compiler implementations when passing floating-point values on the stack. Use `A_GIMME` for more than four arguments or with multiple floating-point arguments.

You can also specify that missing arguments to a message be filled in with default values before your C function receives them. `A_DEFLONG` will put a 0 in place of a missing long argument, `A_DEFFLOAT` will put 0.0 in place of a missing float argument, and `A_DEFSYM` will put the empty symbol (equal to `gensym("")`) in place of a missing symbol argument.

## 5.2 Writing A\_GIMME Functions

A method that uses `A_GIMME` is declared as follows:

```
void myobject_message(t_myobject **x, t_symbol *s, long argc, t_atom *argv);
```

The symbol argument `s` is the message selector. Ordinarily this might seem redundant, but it is useful for the "anything" method as we'll discuss below.

`argc` is the number of atoms in the `argv` array. It could be 0 if the message was sent without arguments. `argv` is the array of atoms holding the arguments.

For typed messages, the atoms will be of type [A\\_SYM](#), [A\\_FLOAT](#), or [A\\_LONG](#). Here is an example of a method that merely prints all of the arguments.

```
void myobject_printargs(t_myobject **x, t_symbol *s, long argc, t_atom *argv)
{
    long i;
    t_atom *ap;
    post("message selector is %s", s->s_name);
    post("there are %ld arguments", argc);
    // increment ap each time to get to the next atom
    for (i = 0, ap = argv; i < argc; i++, ap++) {
        switch (atom_gettype(ap)) {
            case A_LONG:
                post("%ld: %ld", i+1, atom_getlong(ap));
                break;
            case A_FLOAT:
                post("%ld: %.2f", i+1, atom_getfloat(ap));
                break;
            case A_SYM:
                post("%ld: %s", i+1, atom_getsym(ap)->s_name);
                break;
            default:
                post("%ld: unknown atom type (%ld)", i+1, atom_gettype(ap));
                break;
        }
    }
}
```

You can interpret the arguments in whatever manner you wish. You cannot, however, modify the arguments as they may be about to be passed to another object.

## 5.3 Writing "Anything" Methods

As previously mentioned, your object can define a special method bound to the symbol "anything" that will be invoked if no other match is found for a selector. For example:

```
class_addmethod(c, (method)myobject_anything, "anything", A_GIMME, 0);
```

Your function definition for an anything method follows the same pattern as for all other [A\\_GIMME](#) methods:

```
void myobject_anything(t_myobject **x, t_symbol *s, long argc, t_atom *argv)
{
    object_post( (t_object*)x,
                 "This method was invoked by sending the '%s' message to this object.",
                 s->s_name);
    // argc and argv are the arguments, as described in above.
}
```



## Chapter 6

# The Scheduler

The Max scheduler permits operations to be delayed until a later time.

It keeps track of time in double-precision, but the resolution of the scheduler depends on the user's environment preferences. The scheduler also works in conjunction with a low-priority queue, which permits time-consuming operations that might be initiated inside the scheduler to be executed in a way that does not disrupt timing accuracy.

Most objects interface with the scheduler via a clock (#t\_clock) object. A clock is associated with a task function that will execute when the scheduler's current time reaches the clock's time. There is also a function called `schedule()` that can be used for one-off delayed execution of a function. It creates a clock to do its job however, so if your object is going to be using the scheduler repeatedly, it is more efficient to store references to the clocks it creates so the clocks can be reused.

The scheduler is periodically polled to see if it needs to execute clock tasks. There are numerous preferences Max users can set to determine when and how often this polling occurs. Briefly:

- The Overdrive setting determines whether scheduler polling occurs in a high-priority timer thread or the main thread
- The Interval setting determines the number of milliseconds elapse between polling the scheduler
- The Throttle setting determines how many tasks can be executed in any particular scheduler poll

Similar Throttle and Interval settings exist for the low-priority queue as well.

For more information refer to the [Timing](#) documentation. While the details might be a little overwhelming on first glance, the important point is that the exact time your scheduled task will execute is subject to variability. Max permits this level of user control over the scheduler to balance all computational needs for a specific application.

## 6.1 Creating and Using Clocks

There are five steps to using a clock in an external object.

1. Add a member to your object's data structure to hold a pointer to the clock object

```
typedef struct _myobject
{
    t_object m_obj;
    void *m_clock;
} t_myobject;
```

2. Write a task function that will do something when the clock is executed. The function has only a single argument, a pointer to your object. The example below gets the current scheduler time and prints it.

```
void myobject_task(t_myobject *x)
{
    double time;
    sched_gettime(&time);
    post("instance %lx is executing at time %.2f", x, time);
}
```

1. In your new instance routine, create the clock (passing a pointer to your object and the task function) and store the result in your object's data structure.

```
x->m_clock = clock_new((t_object *)x, (method)myobject_task);
```

2. Schedule your clock. Use `clock_fdelay()` to schedule the clock in terms of a delay from the current time. Below we schedule the clock to execute 100 milliseconds from now.

```
clock_fdelay(x->m_clock, 100.);
```

If you want to cancel the execution of a clock for some reason, you can use `clock_unset()`.

```
clock_unset(x->m_clock);
```

1. In your object's free routine, free the clock

```
object_free(x->m_clock);
```

Note that if you call `clock_delay()` on a clock that is already set, its execution time will be changed. It won't execute twice.

## 6.2 Creating and Using Qelems

A qelem ("queue element") is used to ensure that an operation occurs in the low-priority thread. The task function associated with a `#t_qelem` is executed when the low-priority queue is serviced, always in the main (user interface) thread. Any qelem that is "set" belongs to the low-priority queue and will be executed as soon as it serviced.

There are two principal things you want to avoid in the high priority thread: first, time-consuming or unpredictable operations such as file access, and second, anything that will block execution for any length of time – for example, showing a dialog box (including a file dialog).

The procedure for using a qelem is analogous to that for using a clock.

1. Add a member to your object's data structure to hold a pointer to the qelem

```
typedef struct _myobject
{
    t_object m_obj;
    void *m_qelem
} t_myobject;
```

2. Write a task function that will do something when the qelem is executed. The function has only a single argument, a pointer to your object.

```
void myobject_qtask(t_myobject *x)
{
    post("I am being executed a low priority!"
}
```

3. In your new instance routine, create the qelem (passing a pointer to your object and the task function) and store the result in your object's data structure.

```
x->m_qelem = qelem_new((t_object *)x, (method)myobject_qtask);
```

4. Set the qelem by using `qelem_set()`. You could, for example, call `qelem_set()` in a clock task function or in direct response to a message such as bang or int.

```
qelem_set(x->m_qelem);
```

If you want to cancel the execution of a qelem for some reason, you can use `qelem_unset()`.

```
qelem_unset(x->m_qelem);
```

1. In your object's free routine, call `qelem_free()`. Do not call `object_free()` or `freeobject()` – unlike the clock, the qelem is not an object.

```
qelem_free(x->m_qelem);
```

Note that if you call `qelem_set()` on a qelem that is already set, it won't execute twice. This is a feature, not a bug, as it permits you to execute a low-priority task only as fast as the low-priority queue operates, not at the high-priority rate that the task might be triggered. An example would be that a number box will redraw more slowly than a counter that changes its value. This is not something you need to worry about, even if you are writing UI objects, as Max handles it internally (using a qelem).

## 6.3 Defer

The defer function and its variants use a qelem to ensure that a function executes at low-priority. There are three variants: `defer()`, `defer_low()`, and `defer_medium()`. The difference between using `defer()` and a qelem is that `defer()` is a one-shot deal – it creates a qelem, sets it, and then gets rid of it when the task function has executed. The effect of this is that if you have some rapid high-priority event that needs to trigger something to happen at low-priority, `defer()` will ensure that this low-priority task happens every time the high-priority event occurs (in a 1:1 ratio), whereas using a qelem will only run the task at a rate that corresponds to the service interval of the low-priority queue. If you repeatedly `defer()` something too rapidly, the low-priority queue will become backlogged and the responsiveness of the UI will suffer.

A typical use of `defer()` is if your object implements a read message to ask the user for a file. Opening the dialog in the timer thread and waiting for user input will likely crash, but even if it didn't, the scheduler would effectively stop.

To use `defer()`, you write a deferred task function that will execute at low priority. The function will be passed a pointer to your object, plus a symbol and atom list modeled on the prototype for an anything method. You need not pass any arguments to the deferred task if you don't need them, however.

```
void myobject_deferredtask(t_myobject *x, t_symbol *s, long argc, t_atom *argv)
{
    post("I am deferred");
}
```

To call the task, use `defer()` as shown below. The first example passes no arguments. The second passes a couple of long atoms.

```
defer((t_object *)x, (method)myobject_deferredtask, NULL, 0, NULL);
t_atom av[2];
atom_setlong(av, 1);
atom_setlong(av+ 2, 74);
defer((t_object *)x, (method)myobject_deferredtask, NULL, 2, av);
```

Defer copies any atoms you pass to newly allocated memory, which it frees when the deferred task has executed.

### 6.3.1 Defer Variants

defer has two variants, `defer_low()` and `defer_medium()`. Here is a comparison:

#### `defer()`

If executing at high priority, `defer()` puts the deferred task at the front of the low-priority queue. If not executing at high priority, `defer()` calls the deferred task immediately.

#### `defer_low()`

At all priority levels, `defer_low()` puts the deferred task at the back of the low-priority queue.

#### `defer_medium()`

If executing at high priority, `defer_medium()` puts the deferred task at the back of the low-priority queue. If not executing at high priority, `defer_medium()` calls the deferred task immediately.

## 6.4 Schedule

The `schedule()` function is to clocks as `defer()` is to qelems. Schedule creates a clock for a task function you specify and calls `clock_fdelay()` on it to make the task execute at a desired time. As with `defer()`, `schedule()` can copy arguments to be delivered to the task when it executes.

A `schedule()` variant, `schedule_defer()`, executes the task function at low priority after a specified delay.

## Chapter 7

# Memory Allocation

The Max API offers cross-platform calls memory management.

There are two types of calls, those for pointers and those for handles. Handles are pointers to pointers, and were used in the early Mac OS to permit memory to be relocated without changing a reference, and many Mac OS API calls used handle. There are a few legacy Max API calls that use handles as well, but in general, unless the OS or Max requires the use of a handle, you're probably better off using the simpler pointer.

Longtime Max object programmers may have used memory calls `getbytes()` and `freebytes()` in the past, but all memory calls now use same underlying OS mechanisms, so while `getbytes()` and `freebytes()` are still supported, they are restricted to 32K of memory or less due to the arguments they use, and we recommend the use of `sysmem_newptr()` and `sysmem_freeptr()` instead.

Here are some examples of allocating and freeing pointers and handles.

```
char *ptr;
char **hand;
ptr = sysmem_newptr(2000);
post("I have a pointer %lx and it is %ld bytes in size",ptr, sysmem_ptrsize(ptr));
ptr = sysmem_resizeptrclear(ptr, 3000);
post("Now I have a pointer %lx and it is %ld bytes in size",ptr, sysmem_ptrsize(ptr));
sysmem_freeptr(ptr);
hand = sysmem_newhandle(2000);
post("I have a handle %lx and it is %ld bytes in size",hand, sysmem_handlesize(hand));
sysmem_resizehandle(hand, 3000);
post("Now the handle %lx is %ld bytes in size",hand, sysmem_ptrsize(hand));
sysmem_freehandle(hand);
```



# Chapter 8

## Anatomy of a MSP Object

An MSP object that handles audio signals is a regular Max object with a few extras.

Refer to the simplemsp~ example project source as we detail these additions. simplemsp~ is simply an object that adds a number to a signal, identical in function to the regular MSP +~ object if you were to give it an argument of 1.

Here is an enumeration of the basic tasks:

### 8.1 Additional Header Files

After including ext.h and ext\_obex.h, include z\_dsp.h

```
#include "z_dsp.h"
```

### 8.2 C Structure Declaration

The C structure declaration must begin with a `t_pxobject`, not a `t_object`:

```
typedef struct _mydspobject
{
    t_pxobject m_obj;
    // rest of the structure's fields
} t_mydspobject;
```

### 8.3 Initialization Routine

When creating the class with `class_new()`, you must have a free function. If you have nothing special to do, use `dsp_free()`, which is defined for this purpose. If you write your own free function, the first thing it should do is call `dsp_free()`. This is essential to avoid crashes when freeing your object when audio processing is turned on.

```
c = class_new("mydspobject", (method)mydspobject_new, (method)dsp_free, sizeof(t_myspobject), NULL, 0);
```

After creating your class with `class_new()`, you must call `class_dspinit()`, which will add some standard method handlers for internal messages used by all signal objects.

```
class_dspinit(c);
```

Your signal object needs a method that is bound to the symbol "dsp" – we'll detail what this method does below, but the following line needs to be added while initializing the class:

```
class_addmethod(c, (method)mydspobject_dsp64, "dsp64", A_CANT, 0);
```

## 8.4 New Instance Routine

The new instance routine must call `dsp_setup()`, passing a pointer to the newly allocated object pointer plus a number of signal inlets the object will have. If the object has no signal inlets, you may pass 0. The simplemsp~ object (as an example) has a single signal inlet:

```
dsp_setup((t_pxobject *)x, 1);
```

`dsp_setup()` will make the signal inlets (as proxies) so you need not make them yourself.

If your object will have audio signal outputs, they need to be created in the new instance routine with `outlet_new()`. However, you will never access them directly, so you don't need to store pointers to them as you do with regular outlets. Here is an example of creating two signal outlets:

```
outlet_new((t_object *)x, "signal");
outlet_new((t_object *)x, "signal");
```

## 8.5 The DSP Method and Perform Routine

The `dsp64` method specifies the signal processing function your object defines along with its arguments. Your object's `dsp64` method will be called when the MSP signal compiler is building a sequence of operations (known as the DSP Chain) that will be performed on each set of audio samples. The operation sequence consists of a pointers to functions (called perform routines) followed by arguments to those functions.

The `dsp64` method is declared as follows:

```
void mydspobject_dsp64(t_mydspobject **x, t_object *dsp64, short *count, double samplerate, long maxvectorsize,
                        long flags);
```

To add an entry to the DSP chain, your `dsp64` method uses the "dsp\_add64" method of the DSP chain. The `dsp_add64` method is passed an a pointer to your object, a pointer to a perform64 routine that calculates the samples, an optional flag which may alter behavior, and a generic pointer which will be passed on to your perform routine.

```
object_method(dsp64, gensym("dsp_add64"), x, mydspobject_perform64, 0, NULL);
```

The perform routine is not a "method" in the traditional sense. It will be called within the callback of an audio driver, which, unless the user is employing the Non-Real Time audio driver, will typically be in a high-priority thread. Thread protection inside the perform routine is minimal. You can use a clock, but you cannot use qelems or outlets.

Here is a perform routine that adds a constant of 1 to any incoming signal:

```
void mydspobject_perform64(t_mydspobject **x, t_object *dsp64, double **ins, long numins, double **outs, long
                           numouts, long sampleframes, long flags, void *userparam)
{
    double    *in = ins[0];      // first inlet
    double    *out = outs[0];    // first outlet
    int       n = sampleframes; // vector size
    t_double  value;
    while (n--) {              // perform calculation on all samples
        value = *in++;
        *out++ = value + 1.0;
    }
}
```

## 8.6 Free Function

The free function for the class must either be `dsp_free()` or it must be written to call `dsp_free()` as shown in the example below:

```
void mydspobject_free(t_mydspobject **x)
{
    dsp_free((t_pxobject *)x);
    // can do other stuff here
}
```

## Chapter 9

# Advanced Signal Object Topics

Here are some techniques for implementing additional features found in most signal objects.

### 9.1 Saving Internal State

To implement unit generators such as filters and ramp generators, you need to save internal state between calls to your object's perform routine. Here is a very simple low-pass filter (it just averages successive samples) that saves the value of the last sample in a vector to be averaged with the first sample of the next vector. First we add a field to our data structure to hold the value:

```
typedef struct _myfilter
{
    t_pxobject f_obj;
    double      f_sample;
} t_myfilter;
```

Then, in our dsp method, we pass a pointer to the object as one of the DSP chain arguments. The dsp method also initializes the value of the internal state, to avoid any noise when the audio starts.

```
void myfilter_dsp64(t_myfilter *x, t_object *dsp64, short *count, double samplerate, long maxvectorsize, long flags)
{
    object_method(dsp64, gensym("dsp_add64"), x, myfilter_perform64, 0, NULL);
    x->f_sample = 0.0;
}
```

Here is the perform routine, which obtains the internal state before entering the processing loop, then stores the most recent value after the loop is finished.

```
void myfilter_perform64(t_myfilter **x, t_object *dsp64, double **ins, long numins, double **outs, long numouts,
                        long sampleframes, long flags, void *userparam)
{
    double    *in = ins[0];           // first inlet
    double    *out = outs[0];         // first outlet
    int       n = sampleframes;      // vector size
    double    samp = x->f_sample;   // read from internal state
    double    val;
    while (n--) {
        val = *in++;
        *out++ = (val + samp) * 0.5;
        samp = val;
    }
    x->f_sample = samp;            // save to internal state
}
```

## 9.2 Using Connection Information

The third argument to the dsp method is an array of numbers that enumerate the number of objects connected to each of your objects inputs and outputs. More advanced dsp methods can use this information for optimization purposes. For example, if you find that your object has no inputs or outputs, you could avoid calling 'dsp\_add64' altogether. The MSP signal operator objects (such as `+~` and `*~`) use this to implement a basic polymorphism: they look at the connections count to determine whether the perform routine should use scalar or signal inputs. For example, if the right input has no connected signals, the user can add a scalar value sent to the right inlet.

To implement this behavior, you have a few different options. The first option is to write two different perform methods, one which handles the two-signal case, and one which handles the scalar case. The dsp method looks at the count array and passes a different function to dsp\_add64.

```
if (count[1]) // signal connected to second inlet
    object_method(dsp64, gensym("dsp_add64"), x, mydspobject_twosigperform64, 0, NULL);
else
    object_method(dsp64, gensym("dsp_add64"), x, mydspobject_scalarperform64, 0, NULL);
```

The second option is to store the value of the count array for a particular signal in your object's struct. Then the perform method can make the decision whether to use the signal value or a scalar value that has been stored by the object. In this case, many objects use a single sample value from the signal as a substitute for the scalar. Using the first sample (i.e., the value at index 0) is a technique that works for any vector size, since vector sizes could be as small as a single sample. Here is an example of this technique for an object that has two inputs and one output. The connection count for the right input signal is stored in a struct member named `m_count`:

```
x->m_count = count[1];
object_method(dsp64, gensym("dsp_add64"), x, mydspobject_perform64, 0, NULL);
```

Here is a perform routine that uses the connection count information as passed in the format shown above:

```
void mydspobject_perform64(t_mydspobject **x, t_object *dsp64, double **ins, long numins, double **outs, long
    numouts, long sampleframes, long flags, void *userparam)
{
    t_mydspobject *x = (t_mydspobject *)w[1];
    int connected = x->m_count;
    double *in = ins[0];
    double *in2 = ins[1];
    double *out = outs[0];
    int n = sampleframes;
    double in2value;
    // get scalar sample or use signal depending on whether signal is connected
    in2value = connected ? *in2 : x->m_scalarvalue;
    // do calculation here
    // ...
}
```

## 9.3 Working with Buffer Objects

To access a named buffer~ object for either reading or writing sample values, refer to the [Buffers](#) reference.

## Chapter 10

# Sending Messages, Calling Methods

Max objects, such as the one you write, are C data structures in which methods are dynamically bound to functions.

Your object's methods are called by Max, but your object can also call methods itself. When you call a method, it is essential to know whether the method you are calling is **typed** or not.

Calling a typed method requires passing arguments as an array of atoms. Calling an untyped method requires that you know the exact arguments of the C function implementing the method. In both cases, you supply a symbol that names the method.

In the typed method case, Max will take the array of atoms and pass the arguments to the object according to the method's argument type specifier list. For example, if the method is declared to have an argument type specifier list of [A\\_LONG](#), 0, the first atom in the array you pass will be converted to an int and passed to the function on the stack. If there are no arguments supplied, invoking a typed method that has [A\\_LONG](#), 0 as an argument type specifier will fail. To make typed method calls, use [object\\_method\\_typed\(\)](#) or [typedmess\(\)](#).

In the untyped method case, Max merely does a lookup of the symbol in the object, and, if a matching function is found, calls the function with the arguments you pass.

Certain methods you write for your object, such as the assist method for describing your object and the DSP method in audio objects, are declared as untyped using the [A\\_CANT](#) argument type specifier. This means that Max will not typecheck the arguments you pass to these methods, but, most importantly, a user cannot hook up a message box to your object and send it a message to invoke an untyped method. (Try this for yourself – send the assist message to a standard Max object.)

When you use an outlet, you're effectively making a typed method call on any objects connected to the outlet.

## 10.1 Attributes

Attributes are descriptions of data in your object. The standardization of these descriptions permits Max to provide a rich interface to object data, including the pattr system, inspectors, the quick reference menu, @ arguments, etc.

It is essential that you have some understanding of attributes if you are going to write a UI object. But non-UI objects can make use of attributes as well. The discussion below is not specific to UI objects. It does however, use the recently introduced system of macros in ext\_obex\_util.h (included in ext\_obex.h) for defining attributes, as well as describing them using attributes of attributes (attr attrs). You can read more detailed descriptions of the underlying attribute definition mechanisms on a per-function basis in the [Attributes](#) reference.

### 10.1.1 Attribute Basics

While attributes can be defined for a specific instance of an object, it's much more common to define an attribute for a class. In such a case, each instance of the class will have the attribute description, but the value will be instance specific. The discussion here focuses only on class attributes.

When an attribute is declared and is made user-settable, a user can send a message to your object consisting of the attribute name and arguments that represent the new value of the attribute. For example, if you declare an attribute called trackcount, the message trackcount 20 will set it to 20. You don't need to do anything special to obtain this behavior. In addition, user-settable attributes will appear when the user opens the inspector on your object.

If you define your attribute as an offset attribute, you describe its location (and size) within your object's C data structure. Max can then read and write the data directly. You can also define custom getter and setter routines if the attribute's value is more complex than simply a stored number. As a theoretical example, you could have an object with an attribute representing the Earth's population. If this value was not able to be stored inside your object, your custom getter routine could initiate a global census before returning the result. A custom setter for the earth's population might do something nasty if the value was set to zero. If you are not a misanthrope, you can take advantage of the ability to set such an attribute to be read-only.

### 10.1.2 Defining Attributes

Attributes are defined when you are defining methods in your initialization routine. You can define your attributes before your methods if you like, but by convention, they are typically defined after the methods. For each definition, you'll specify the name, size, and offset of the corresponding member in your object's data structure that will hold the data. For example, let's say we have an object defined as follows:

```
typedef struct _myobject {
    t_object m_obj;
    long m_targetaddress;
    t_symbol *m_shipname;
    char m_compatmode;
} t_myobject;
```

We want to create attributes for m\_targetaddress, m\_shipname, and m\_compatmode. For each data type (and a few others), there are macros in ext\_obex\_util.h that will save a fair amount of typing. So, for example, we can define an attribute for m\_targetaddress that uses CLASS\_ATTR\_LONG. Here are attribute definitions for all of the members of our data structure above.

```
CLASS_ATTR_LONG(c, "targetaddress", 0, t_myobject, m_targetaddress);
CLASS_ATTR_SYM(c, "shipname", 0, t_myobject, m_shipname);
CLASS_ATTR_CHAR(c, "compatibilitymode", 0, t_myobject, m_compatmode);
```

### 10.1.3 Attributes With Custom Getters and Setters

In some cases, it is not enough to have Max read and write data in your object directly. In some cases (as in the world population example above) you may have data you need to calculate before it can be returned as a value. In other cases, you may need to do something to update other object state when an attribute value changes. To handle these challenges, you can define custom attribute getter and setter routines. The getter will be called when the value of your attribute is accessed. The setter will be called when someone changes the value of your attribute.

As an example, suppose we have an object that holds onto an array of numbers, and we want to create an attribute for the size of the array. Since we'll want to resize the array when the attribute value changes, we will define a custom setter for our attribute. The default getter is adequate if we store the array size in our object, but since we want to illustrate how to write an attribute getter, we'll write the code so that the array size is computed from the size of the memory pointer we allocate. First, here is our object's data structure:

```
typedef struct _myobject {
    t_object m_obj;
    long *m_data;
} t_myobject;
```

We also have prototypes for our custom attribute setter and getter:

```
t_max_err myobject_size_get(t_myobject **x, t_object *attr, long *argc, t_atom **argv);
t_max_err myobject_size_set(t_myobject **x, t_object *attr, long argc, t_atom *argv);
```

Here is how we define our attribute using `CLASS_ATTR_ACCESSORS` macro to define the custom setter and getter. Because we aren't really using an "offset" due to the custom setter and getter, we can pass any data structure member as a dummy. (Only the default attribute getter and setter will use this offset, and they are out of the picture.)

```
CLASS_ATTR_LONG(c, "size", 0, t_myobject, m_obj);
CLASS_ATTR_ACCESSORS(c, "size", myobject_size_get, myobject_size_set);
```

Now, here is an implementation of the custom setter for the array size. For the setter, we use the handy Max API function `sysmem_resizeptr` so we can effectively "resize" our array and copy the data into it in one step. The setter uses atoms, so we have to obtain the value from the first item in the argv array.

```
t_max_err myobject_size_set(t_myobject **x, t_object *attr, long argc, t_atom *argv)
{
    long size = atom_getlong(argv);
    if (size < 0)          // bad size, don't change anything
        return 0;
    if (x->m_data)
        x->m_data = (long *)sysmem_resizeptr((char *)x->m_data, size * sizeof(long));
    else                  // first time alloc
        x->m_data = (long *)sysmem_newptr(size * sizeof(long));
    return 0;
}
```

The getter also uses atoms for access, but we are returning a pointer to an array of atoms. The caller of the getter has the option to pre-allocate the memory (passing in the length in argc and the pointer to the memory in argv) or pass in 0 for argc and set the contents of argv to NULL and have the getter allocate the memory. The easiest way to handle this case is to call the utility function `atom_alloc`, which will figure out what was passed in and allocate memory for a returned atom if necessary.

```
t_max_err myobject_size_get(t_myobject **x, t_object *attr, long *argc, t_atom **argv)
{
    char alloc;
    long size = 0;
    atom_alloc(argc, argv, &alloc);      // allocate return atom
    if (x->m_data)
        size = sysmem_ptrsize((char *)x->m_data) / sizeof(long); // calculate array size based on ptr size
    atom_setlong(*argv, size);
    return 0;
}
```

## 10.2 Receiving Notifications

As an alternative to writing a custom setter, you can take advantage of the fact that objects receive a "notify" message whenever one of their attributes is changed. The prototype for a notify method is as follows:

```
t_max_err myobject_notify(t_myobject **x, t_symbol *s, t_symbol *msg, void *sender, void *data);
```

Add the following to your class initialization so your notification method will be called:

```
class_addmethod(c, (method)myobject_notify, "notify", A_CANT, 0);
```

The notify method can handle a variety of notifications (more documentation on this is coming soon!), but the one we're interested in is "attr\_modified" – the notification type is passed to the notify method in the msg argument. Here is an example of a notify method that prints out the name of the attribute that has been modified. You could take any action instead. To obtain the name, we interpret the data argument to the notify method as an attribute object. As an attribute is a regular Max object, we can use `object_method` to send it a message. In the case we are sending the message `getname` to the attribute object to obtain its name.

```
t_max_err myobject_notify(t_myobject **x, t_symbol *s, t_symbol *msg, void *sender, void *data)
```

```
{  
    t_symbol *attrname;  
    if (msg == gensym("attr_modified")) {      // check notification type  
        attrname = (t_symbol *)object_method((t_object *)data, gensym("getname"));      // ask attribute object  
        for name  
            object_post((t_object *)x, "changed attr name is %s", attrname->s_name);  
    }  
    return 0;  
}
```

# Chapter 11

## Anatomy of a UI Object

Max user interface objects are more complex than normal non-user-interface objects.

If you have nothing in particular to display, or do not need to create a unique interface for user interaction or editing, it would be better to avoid writing one. However, if you want the details, we have them for you!

In order to create a user interface object, you'll need to be familiar with [Attributes](#), as they are used extensively. If you examine a toggle object in the inspector in Max, you will see a few attributes that have been defined as belonging to the toggle class, namely:

- Background Color
- Check Color
- Border Color

We'll show how attributes are defined and described so that the inspector can edit them properly.

In addition to attributes, user interface objects draw in a box and respond to user events such as mouse clicks and keyboard events. We'll show how to implement drawing an object's paint method as well user interaction in the mousedown, mousedrag, and mouseup methods.

This chapter only covers basic drawing of lines and filled rectangles. But you can take advantage of a complete graphics API called jgraphics, intended to be used in a user interface object's paint method. We discuss [JGraphics](#) in more detail in a separate chapter. You may also find the jgraphics.h header file descriptions of the set of functions helpful.

The SDK examples contain two user interface projects – the one we'll discuss in this chapter is called *uisimp* and is a version of the toggle object with a more complicated check box and user interaction. The second project is called *pictmeter~*, a more advanced object that uses audio as well as image files.

The *uisimp* object differs from the toggle object in a couple of ways:

- it tracks the mouse even when it isn't down and "looks excited" when the mouse passes over it
- it tracks the mouse while the user is holding the mouse down to show a sort of "depressed" appearance when turning the toggle on

- the new toggle state value is sent out when the mouse is released rather than when the mouse is down. In addition, the uisimp object tracks the mouse and does not change the state if the mouse is released outside of the object's box
- it doesn't have rounded corners
- it has a solid square for a "checked state" instead of an X

Otherwise, it acts largely as the toggle does.

The first thing we suggest you do is build the uisimp object and test it out. Once the object is properly building, type "uisimp" into an object box and you can try it out.

## 11.1 Required Headers

UI objects require that you include two header files, jpatcher\_api.h and jgraphics.h:

```
#include "jpatcher_api.h"
#include "jgraphics.h"
```

The header file jpatcher\_api.h includes data structures and accessor functions required by UI objects. The header file jgraphics.h includes data structures and functions for drawing.

## 11.2 UI Object Data Structure

The first part of a UI object is a [t\\_jbox](#), not a [t\\_object](#). You should generally avoid direct access to fields of a [t\\_jbox](#), particularly when changing values, and use the accessor functions defined in jpatcher\_api.h. For example, if you change the rectangle of a box without using the accessor function [jbox\\_set\\_rect\(\)](#), the patcher will not be notified properly and the screen will not update.

Following the [t\\_jbox](#), you can add other fields for storing the internal state of your object. In particular, if you are going to be drawing something using color, you will want to create attributes that reference fields holding colors in your object. We'll show you how to do this below. Here is the declaration of the [t\\_uisimp](#) data structure.

```
typedef struct _uisimp
{
    t_jbox u_box;                                // header for UI objects
    void *u_outlet;                             // outlet pointer
    long u_state;                               // state (1 or 0)
    char u_mouseover;                           // is mouse over the object
    char u_mousedowninside;                     // is mouse down within the object
    char u_trackmouse;                          // if non-zero, track mouse when button not down
    t_jrgba u_outline;                          // outline color
    t_jrgba u_check;                            // check (square) color
    t_jrgba u_background;                      // background color
    t_jrgba u_hilite;                           // highlight color (when mouse is over and when clicking to check box)
} t_uisimp;
```

The [t\\_jrgba](#) structure defines a color with four doubles for red, green, blue, and alpha. Each component ranges from 0-1. When red, green, and blue are all 0, the color is black; when red, green, and blue are 1, the color is white. By defining color attributes using [t\\_jrgba](#) structures, you will permit the user to use the standard color picker from the inspector to configure colors for your object.

The structure members [u\\_mouseover](#) and [u\\_mousedowninside](#) are used to signal the code that paints the toggle from the code that handles mouse interaction. We'll discuss this more in the "interaction strategy" section below.

## 11.3 Initialization Routine for UI Objects

Once you've declared your object's struct, you'll write your initialization ( `ext_main()` ) routine to set up the class, declaring methods and attributes used by UI objects.

The first addition to the class initialization of a normal Max object you need to make is a call to `jbox_initclass()`. This adds standard methods and attributes common to all UI objects. Here's how you should to it:

```
c = class_new("uisimp", (method)uisimp_new, (method)uisimp_free, sizeof(t_uisimp), 0L, A_GIMME, 0);
c->c_flags |= CLASS_FLAG_NEWDICTIONARY;
jbox_initclass(c, JBOX_FIXWIDTH | JBOX_COLOR);
```

The line `c->c_flags |= CLASS_FLAG_NEWDICTIONARY` is required, but the flags passed to `jbox_initclass` – `JBOX_FIXWIDTH` and `JBOX_COLOR` – are optional. `JBOX_FIXWIDTH` means that when your object is selected in a patcher, the Fix Width menu item will be enabled to resize your object to its class's default dimensions. We'll specify the default dimensions in a moment. `JBOX_COLOR` means that your object will be given a color attribute so that it can be edited with the color picked shown by the Color... menu item. This is a way to edit a "basic" color of your object without opening the inspector. If neither of these behaviors apply to your object, feel free to pass 0 for the flags argument to `jbox_initclass()`.

## 11.4 UI Object Methods

Next we need to bind a few standard methods. The only required method for UI objects is `paint`, which draws the your object's content when its box is visible and needs to be redrawn.

```
class_addmethod(c, (method)uisimp_paint, "paint", A_CANT, 0);
```

We'll discuss the `paint` method in detail below. It makes use of the `JGraphics` API, which is described in more detail in its own chapter.

Our `uisimp` toggle will respond to mouse gestures, so we will define a set of mouse handling methods.

```
class_addmethod(c, (method)uisimp_mousedown, "mousedown", A_CANT, 0);
class_addmethod(c, (method)uisimp_mousedrag, "mousedrag", A_CANT, 0);
class_addmethod(c, (method)uisimp_mouseup, "mouseup", A_CANT, 0);
class_addmethod(c, (method)uisimp_mouseenter, "mouseenter", A_CANT, 0);
class_addmethod(c, (method)uisimp_mouseleave, "mouseleave", A_CANT, 0);
class_addmethod(c, (method)uisimp_mousemove, "mousemove", A_CANT, 0);
class_addmethod(c, (method)uisimp_mousewheel, "mousewheel", A_CANT, 0);
```

`mousedown` is sent to your object when the user clicks on your object – in other words, when the mouse is moved over the object and the primary mouse button is depressed. `mousedrag` is sent after an initial `mousedown` when the mouse moves and the button is still held down from the click. `mouseup` is sent when the mouse button is released after a `mousedown` is sent. `mouseenter` is sent when the mouse button is not down and the mouse moves into your object's box. `mousemove` is sent – after a `mouseenter` – when the mouse button is not down but the mouse position changes inside your object's box. `mouseleave` is sent when the mouse button is not down and the mouse position moves from being over your object's box to being outside of it. `mousewheel` is sent when information about the scrollwheel on the mouse (or scrolling from another source such as a trackpad) is transmitted while the cursor is hovering over your object.

You are not obligated to respond to any of these messages. You could, for example, only respond to `mousedown` and ignore the other messages.

It might be helpful to summarize mouse messages in the following "rules" (although normally it's not necessary to think about them explicitly):

- `mousedown` will always be followed by `mouseup`, but not necessarily by `mousedrag` if the button press is rapid and there is no movement while the mouse button is pressed.

- mouseenter will always be followed by mouseleave, but
- mouseenter will always precede mousemove
- mouseleave will be sent only after a mouseenter is sent
- You cannot count on any particular relationship between the mousedown / mousedrag / mouseup sequence and the mouseenter / mousemove / mouseleave sequence.

We'll look at the actual implementation of mouse handling methods below.

## 11.5 Defining Attributes

After the declaration of standard methods, your object will define its own attributes. By using what we call "attribute attributes" you can further describe attributes so that they can be appropriately displayed and edited in the inspector as well as saved in a patcher (or not). You can also set default values for attributes that are automatically copied to your object when it is instantiated, and mark an attribute so that your object is redrawn when its value changes.

As a convenience, we've defined a series of macros in ext\_obex\_util.h (which is included when your object includes ext\_obex.h) that reduce the amount of typing needed to define attributes and attribute attributes.

Most UI object attributes are offset attributes; that is, they reference a location in your object's data structure by offset and size. As an example, uisimp has a char offset attribute called trackmouse that specifies whether the object will change the object's appearance when the mouse moves over it. Here's how this is defined:

```
CLASS_ATTR_CHAR(c, "trackmouse", 0, t_uisimp, u_trackmouse);
CLASS_ATTR_STYLE_LABEL(c, "trackmouse", 0, "onoff", "Track Mouse");
CLASS_ATTR_SAVE(c, "trackmouse", 0);
```

The first line, `CLASS_ATTR_CHAR`, defines a char-sized offset attribute. If you look at the declaration of `t_uisimp`, you can see that the `u_trackmouse` field is declared to be a char. The `CLASS_ATTR_CHAR` macro takes five arguments.

- The first argument is the class for which the attribute is being declared.
- The second argument is the name of the attribute. You can use send a message to your object with this name and a value and set the attribute.
- The third argument is a collection of attribute flags. For the attributes (and attribute attributes) we'll be defining in the uisimp object, the flags will be 0, but you can use them to make attributes read-only with `ATTR_SET_OPAQUE_USER`.
- The fourth argument is the name of your object's structure containing the field you want to use for the attribute
- The fifth argument is the field name you want to use for the attribute

The fourth and fifth arguments are used to calculate the offset of the beginning of the field from the beginning of the structure. This allows the attribute to read and write the memory occupied by the field directly.

The second line, `CLASS_ATTR_STYLE_LABEL`, defines some attribute attributes for the trackmouse attribute. This macro takes five arguments as well:

- The first argument is the class for which the attribute attributes are being declared.

- The second argument is the name of the attribute, which should have already been defined by a `CLASS_ATTR_CHAR` or similar attribute declaration
- The third argument is usually 0 – it is an attribute flags argument for the attribute attributes
- The fourth argument is the style of the attribute. "onoff" is used here for a setting in your object that will be a toggle. By using the onoff style the trackmouse attribute will appear with a checkbox in the inspector window. Effectively, this macro defines an attribute called "style" that is attached to the "trackmouse" attribute and set its value to the symbol "onoff" in one step.
- The fifth argument is a string used as a descriptive label for the attribute that appears in the inspector and other places in the Max user interface. If you don't supply a label, the attribute name will be shown. The string is used as the value of a newly created "label" attribute attribute.

The category attribute attribute is used to organize your object's attributes in the inspector window. For the trackmouse attribute, we use the "Behavior" category, and for the color attributes discussed below, we use "Color" – look at the inspector category tabs for a few UI objects that come with Max for suggested standard category names. You're free to create your own.

To define a category for a single attribute, you can use the `CLASS_ATTR_CATEGORY` macro:

```
CLASS_ATTR_CATEGORY(c, "trackmouse", 0, "Behavior");
```

To define a category for a series of attributes, you can use `CLASS_STICKY_ATTR`, which applies the current value of a specified attribute attribute to any attributes subsequently defined, until a `CLASS_STICKY_ATTR_CLEAR` is set for an attribute attribute name. `CLASS_STICKY_ATTR` is used in uisimp to apply the "Color" category to a set of three color attributes.

```
CLASS_STICKY_ATTR(c, "category", 0, "Color");
```

Color attributes are defined using `CLASS_ATTR_RGBA`. The uisimp object defines four color attributes. Here is the first, called `bgcolor`:

```
CLASS_ATTR_RGBA(c, "bgcolor", 0, t_uisimp, u_background);
CLASS_ATTR_DEFAULTNAME_SAVE_PAINT(c, "bgcolor", 0, ".1. 1. 1. 1.");
CLASS_ATTR_STYLE_LABEL(c, "bgcolor", 0, "rgba", "Background Color");
```

The difference between `CLASS_ATTR_RGBA` and `CLASS_ATTR_CHAR` for defining an attribute is that `CLASS_ATTR_RGBA` expects the name of a structure member declared of type `t_jrgba` rather than type char. When set, the attribute will assign values to the four doubles that make up the components of the color.

The next line uses the `CLASS_ATTR_DEFAULTNAME_SAVE_PAINT` macro. This sets three things about the `bgcolor` attribute. First it says that the color attribute `bgcolor` can be assigned a default value via the object defaults window. So, if you don't like the standard white defined by the object, you can assign your own color for the background color of all newly created uisimp objects. The four values 1 1 1 1 supplied as the last argument to `CLASS_ATTR_DEFAULTNAME_SAVE_PAINT` specify the "standard" default value that will be used for the `bgcolor` attribute in the absence of any overrides from the user.

The `SAVE` aspect of this macro specifies that this attribute's values should be saved with the object in a patcher. A patcher file saves an object's class, location and connections, but it can also save the object's appearance or any other attribute value you specify, by using the "save" attribute attribute.

The `PAINT` aspect of this macro provides the ability to have your object redrawn whenever this attribute (`bgcolor`) changes. However, to implement auto-repainting on attribute changes, you'll need to add the following code when initializing your class:

```
class_addmethod(c, (method) jbox_notify, "notify", A_CANT, 0);
```

The function `jbox_notify()` will determine whether an attribute that has caused a change notification to be sent has its `paint` attribute attribute set, and if so, will call `jbox_redraw()`. If you write your own `notify` method because you want to respond to changes in attributes or other environment changes, you \*must\* call `jbox_notify()` inside of it.

### 11.5.1 Standard Color Attribute

At the beginning of our initialization routine, we passed `JBOX_COLOR` as a flag to `jbox_initclass()`. This adds an attribute to our object called `color`, which uses storage provided in the `t_jbox` to keep track of a color for us. The `color` attribute is a standard name for the "most basic" color your object uses, and if you define it, the `Color` menu item in the `Object` menu will be enabled when your object is selected, permitting the user to change the color without opening the inspector.

If you use `JBOX_COLOR`, you don't need to define the `color` attribute using `CLASS_ATTR_RGBA` – `jbox_initclass()` will do it for you. However, the `color` attribute comes unadorned, so you are free to enhance it with attribute attributes. Here's what `uisimp` does:

```
CLASS_ATTR_DEFAULTNAME_SAVE_PAINT(c, "color", 0, "0. 0. 0. 1.");
CLASS_ATTR_STYLE_LABEL(c,"color",0,"rgba","Check Color");
```

### 11.5.2 Setting a Default Size

Another attribute defined for your object by `jbox_initclass()` is called `patching_rect`. It holds the dimensions of your object's box. If you want to set a standard size for new instances of your object, you can give the `patching_rect` a set of default values. Use 0 0 for the first two values (x and y position) and use the next two values to define the width and height. We want a small square to be the default size for `uisimp`, so we use `CLASS_ATTR_DEFAULT` to assign a default value to the `patching_rect` attribute as follows:

```
CLASS_ATTR_DEFAULT(c,"patching_rect",0, "0. 0. 20. 20.");
```

## 11.6 New Instance Routine

The UI object new instance routine is more complicated than that of a normal Max object. Each UI object is passed a `t_dictionary` (a hierarchically structured collection of data accessed by symbolic names) containing the information needed to instantiate an instance. For UI objects, data elements in the dictionary correspond to attribute values. For example, if your object saved an attribute called "bgcolor" you will be able to access the saved value in your new instance routine from the dictionary using the same name `bgcolor`.

If the instance is being created from the object palette or by the typing the name of your object into an object box, the dictionary will be filled in with default values. If the object is being created by reading a patcher file, the dictionary will be filled in with the saved attributes stored in the file. In most cases, you don't need to work with the dictionary directly, unless you've added proprietary non-attribute information to your object's dictionary that you want to look for and extract. However, you do need to pass the dictionary to some standard routines, and initialize everything in the right order.

Let's take a look at the pattern you should follow for your object's new instance routine.

First, the new instance routine is declared as follows:

```
void *uisimp_new(t_symbol *s, long argc, t_atom *argv);
```

We will get the dictionary that defines the object out of the arguments passed in `argc`, `argv`. (The symbol argument `s` is the name of the object.) If obtaining the dictionary fails, we should return `NULL` to indicate we didn't make an instance.

```
void *uisimp_new(t_symbol *s, long argc, t_atom *argv);
{
    t_uisimp *x = NULL;
    t_dictionary *d = NULL;
    long boxflags;
    if (! (d = object_dictionaryarg(argc, argv)))
        return NULL;
```

Next, we allocate a new instance of the object's class:

```
x = (t_uisimp *)object_alloc(s_uisimp_class);
```

Then we need to initialize the options for our box. Our object uses the options that are not commented out.

```
boxflags = 0
| JBOX_DRAWFIRSTIN
| JBOX_NODRAWBOX
| JBOX_DRAWINLAST
| JBOX_TRANSPARENT
// | JBOX_NOGROW
| JBOX_GROWY
// | JBOX_GROWBOTH
// | JBOX_HILITE
// | JBOX_BACKGROUND
| JBOX_DRAWBACKGROUND
// | JBOX_NOFLOATINSPECTOR
// | JBOX_MOUSEDRAGDELTA
// | JBOX_TEXTFIELD
;
```

Here is some more detail about each of the box flags.

We pass the flags along with a pointer to our newly created instance and the argc, argv arguments to `jbox_new()`. The name is a little misleading. `jbox_new()` does not instantiate your box. As we explained above, your UI object has a `t_jbox` at the beginning. `jbox_new()` just initializes the `t_jbox` for you. `jbox_new()` doesn't know about the other stuff in your object's data structure that comes after the `t_jbox`. You'll have to initialize the extra items yourself.

```
jbox_new((t_jbox *)x, boxflags, argc, argv);
```

Once `jbox_new()` has been called, you then assign the `b_firstin` pointer of your `t_jbox` header to point to your object. Essentially this assigns the object that will receive messages from objects connected to your leftmost inlet (as well as other inlets via inlets or proxies you create). This step is easily forgotten and will cause most things not to work until you remember it. `jbox_new()` will obtain the attributes common to all boxes such as the `patching_rect`, and assign them to your object for you.

```
x->u_box.b_firstin = (void *)x;
```

Next, you are free to initialize any members of your object's data structure, as well as declare inlets. These steps are the same for UI objects as for non-UI objects.

```
x->u_mousedowninside = x->u_mouseover = x->u_state = 0;
x->u_out = intout((t_object *)x);
```

Once your object is in a safe initialized state, call `attr_dictionary_process()` if you've defined any attributes. This will find the attributes in the dictionary your object received, then set them to the values stored in the dictionary. There is no way to guarantee the order in which the attributes will be set. If this is a problem, you can obtain the attribute values "by hand" and assign them to your object.

Note that you do not need to call `attr_dictionary_process()` if you have not defined any attributes. `jbox_new()` will take care of setting all attributes common to all UI objects.

```
attr_dictionary_process(x, d);
```

As the last thing to do before returning your newly created UI object, and more specifically after you've initialized everything to finalize the appearance of your object, call `jbox_ready()`. `jbox_ready()` will paint your object, calculate the positions of the inlets and outlets, and perform other initialization tasks to ensure that your box is a proper member of the visible patcher.

If your object does not appear when you instantiate it, you should check whether you do not have a `jbox_ready()` call.

```
jbox_ready((t_jbox *)x);
```

Finally, as with any instance creation routine, the newly created object will be returned.

```
return x;
```

## 11.7 Dynamic Updating

Drawing anything to the screen must be limited to your paint method (this was not the case with the previous UI object API in Max). If you want to redraw something, you need to call `jbox_redraw()` to cause the screen to be redrawn. This is necessary because your object is part of a compositing user interface that must be managed by the patcher as a whole to avoid screen artifacts. The `jbox_redraw()` routine calculates the area of the screen that needs to be redrawn, then informs the Mac or Windows "window manager" to mark this area as invalid. At some later point in time, the OS will invoke the patcher's paint routine, which will dispatch to all of the boxes inside the invalid area according to the current Z-order of all the boxes. Boxes that are in the background are drawn first, so that any transparent or semi-transparent boxes can be drawn on top of them. In addition, unless you specify otherwise, the last drawn image of a box is cached in a buffer, so that your paint method will only be called when you explicitly invalidate your object's content with `jbox_redraw()`. In other words, you can't count on "global patcher drawing" to invoke your paint method.

The basic strategy you'll want to use in thinking about redrawing is that you will set internal state in other methods, then call `jbox_redraw()`. The paint method will read the internal state and adjust its drawing appropriately. You'll see this strategy used in the `uisimp` object as it tracks the mouse.

## 11.8 The Paint Method

Your object's paint method uses the `jgraphics` API to draw. The header file, `jgraphics.h`, provides a description of each of the routines in the API. Here we will only discuss general principles and features of drawing with `uisimp`'s relatively simple paint method. There is also a `jgraphics` example UI object that contains a number of functions showing how various drawing tasks can be performed.

Drawing in Max is resolution-independent. The "size" of your object's rectangle is always the pixel size when the patcher is scaled to 100% regardless of the zoom level, and any magnification or size reduction to the actual screen is automatically handled by matrix transforms. Another thing that is handled automatically for you is drawing to multiple views. If a patcher is invisible (i.e., a subpatcher that has not been double-clicked), it does not have any views. But if it is visible, a patcher can have many patcherviews. If your UI object box is in a patcher with multiple views open, your paint method will be called once for each view, and will be passed different a patcherview object each time. For most objects, this will pose few problems, but for objects to work properly when there are anywhere from zero to ten views open, they cannot change their internal state in the paint method, they can only read it. As an example, if your object had a boolean "painted" field in its structure that would be set when the paint method had finished, it would not work properly in the cases where the box was invisible or where it was shown in multiple patcher views, because it would either be set zero or more than once.

The first step for any paint method is to obtain the `t_jgraphics` object from the patcherview object passed to the paint method. The patcherview is an opaque `t_object` that you will use to access information about your box's rectangle and its graphics context. A patcherview is not the same thing as a patcher; as mentioned above, there could be more than one patcherview for a patcher if it has multiple views open.

```
void uisimp_paint(t_uisimp *x, t_object *patcherview)
{
    t_rect rect;
    t_jgraphics *g = (t_jgraphics*) patcherview_get_jgraphics(patcherview);           // obtain graphics context
```

After obtaining the `t_jgraphics` object, the next thing that you'll need to do is determine the rectangle of your box. A view of a patcher may be in either patching or presentation mode. Since each mode can have its own rectangle, it is necessary to use the patcherview to obtain the rectangle for your object.

```
jbox_get_rect_for_view((t_object *)x, patcherview, &rect);
```

The `t_rect` structure specifies a rectangle using the x and y coordinates of the top left corner, along with the width and height. However, the coordinates of the `t_jgraphics` you'll be using to draw into always begin at 0 for the top left corner, so you'll only care about the width and height, at least for drawing.

The first thing we'll draw is just an outline of our box using the value of the outline color attribute. First we'll set the color we want to use, then make a rectangular path, then finally we'll stroke the path we've made.

With calls such as `jgraphics_rectangle()`, the rectangular shape is added to the existing path. The initial path is empty, and after calling `jgraphics_stroke()` or `jgraphics_fill()`, the path is again cleared. (If you want to retain the path, you can use the `jgraphics_stroke_preserve()` and `jgraphics_fill_preserve` variants.)

```
jgraphics_set_source_jrgba(g, &x->u_outline);
jgraphics_set_line_width(g, 1.);
jgraphics_rectangle(g, 0., 0., rect.width, rect.height);
jgraphics_stroke(g);
```

You do not need to destroy the path before your paint method is finished. This will be done for you, but the fact that the path does not survive after the paint method is finished means you can't make a path and then store it without copying it first. Such a strategy is not recommended in any case, since your object's rectangle might change unpredictably from one paint method invocation to the next, which will likely cause your path to be the wrong shape or size.

The next feature of the paint method is to draw an inner outline if the mouse is moved over the box. Detecting the mouse's presence over the box happens in the `mouseenter` / `mouseleave` methods described below – but essentially, we know that the mouse is over our object if the `u_mouseover` has been set by these mouse tracking methods.

To draw a rectangle that is inset by one pixel from the box rectangle, we use the rectangle starting at 1, 1 with a width of the box width - 2 and a height of the box height - 2.

```
// paint "inner highlight" to indicate mouseover
if (x->u_mouseover && !x->u_mousedowninside) {
    jgraphics_set_source_jrgba(g, &x->u_hilite);
    jgraphics_set_line_width(g, 1.);
    jgraphics_rectangle(g, 1., 1., rect.width - 2, rect.height - 2);
    jgraphics_stroke(g);
}
```

Some similar code provides the ability to show the highlight color when the user is about to check (turn on) the toggle:

```
if (x->u_mousedowninside && !x->u_state) {           // paint hilite color
    jgraphics_set_source_jrgba(g, &x->u_hilite);
    jgraphics_rectangle(g, 1., 1., rect.width - 2, rect.height - 2);
    jgraphics_fill(g);
}
```

Finally, we paint a square in the middle of the object if the toggle state is non-zero to indicate that the box has been checked. Here we are filling a path instead of stroking it. Note also that we use the call `jbox_get_color()` to get the "standard" color of our object that is stored inside the `t_jbox`. As we've specified by using the `JBOX_COLOR` flag for `jbox_initclass()` in our initialization routine, the color obtained by `jbox_get_color()` for the "check" (really just a square of solid color) is the one the user can change with the Color... item in the Object menu.

```
if (x->u_state) {
    t_jrgba col;
    jbox_get_color((t_object *)x, &col);
    jgraphics_set_source_jrgba(g, &col);
    if (x->u_mousedowninside)           // make rect bigger if mouse is down and we are unchecking
        jgraphics_rectangle(g, 3., 3., rect.width - 6, rect.height - 6);
    else
        jgraphics_rectangle(g, 4., 4., rect.width - 8, rect.height - 8);
    jgraphics_fill(g);
}
```

Clearly, a quick perusal of the `jgraphics.h` header file will demonstrate that there is much more to drawing than we've discussed here. But the main purpose of the `uisimp` paint method is to show how to implement "dynamic" graphics that follow the mouse. Now we'll see the mouse tracking side of the story.

## 11.9 Handling Mouse Gestures

When the mouse is clicked, dragged, released, or moved inside its box, your object will receive messages. In the uisimp example we've defined methods for most of the mouse gesture messages available, and we've implemented them to change internal state in the object, then call `jbox_redraw()` to repaint the object to reflect the new state. This strategy produces a "dynamic" appearance of a gadget users associate with a typical graphical interface – in this case a toggle checkbox.

All mouse gesture methods are declared in the same way:

```
void myobject_mouse(t_myobject *x, t_object *patcherview, t_pt pt, long modifiers);
```

Let's first look at the most commonly implemented mouse gesture handler, the mousedown method that responds to an initial click on the object. As you can see, it is very simple; it merely sets `u_mousedowninside` to true, then calls `jbox_redraw()`, causing the box to be repainted. We've defined this toggle not to change the actual state until the mouse is released (unlike the standard Max toggle object), but we do want to give the user some feedback on the initial mouse down that something is going to happen. If you look back at the paint method, you can see that `u_mousedowninside` is used to change the way the object is painted to give it a "pending state change" appearance that will be finalized when the mouse is released inside the box.

```
void uisimp_mousedown(t_uisimp *x, t_object *patcherview, t_pt pt, long modifiers)
{
    x->u_mousedowninside = true;      // wouldn't get a click unless it was inside the box
    jbox_redraw((t_jbox *)x);
}
```

If we test the mouse position to ensure that it is inside the box when it is released, we provide the opportunity for the user to cancel the act of toggling the state of the object by moving the cursor outside of the box before releasing the button. To provide feedback to the user that this is going to happen, we've implemented a mouseshift method that performs this test and redraws the object if the "mouse inside" condition has changed from its previous state. The mouseshift message will be sent to your object as long as the mouse button is still down after an initial click and the cursor has moved, even if the cursor moves outside of the boundaries of your object's box.

Note that, as with the paint method, we use the patcherview to get the current box rectangle. We can then test the point we are given to see if it is inside or outside the box.

```
void uisimp_mouseshift(t_uisimp *x, t_object *patcherview, t_pt pt, long modifiers)
{
    t_rect rect;
    // test to see if mouse is still inside the object
    jbox_get_rect_for_view((t_object *)x, patcherview, &rect);
    // redraw if changed
    if (pt.x >= 0 && pt.x <= rect.width && pt.y >= 0 && pt.y <= rect.height) {
        if (!x->u_mousedowninside) {
            x->u_mousedowninside = true;
            jbox_redraw((t_jbox *)x);
        }
    } else {
        if (x->u_mousedowninside) {
            x->u_mousedowninside = false;
            jbox_redraw((t_jbox *)x);
        }
    }
}
```

Our mouseup method uses the last value of `u_mousedowninside` as the determining factor for whether to toggle the object's internal state. If `u_mousedowninside` is false, no state change happens. But if it is true, the state changes and the new state value is sent out the object's outlet (inside `uisimp_bang()`).

```
if (x->u_mousedowninside) {
    x->u_state = !x->u_state;
    uisimp_bang(x);
    x->u_mousedowninside = false;
    jbox_redraw((t_jbox *)x);
}
```

Finally, we've implemented mouseenter, mousemove, and mouseleave methods to provide another level of "mouse over" style highlighting for the object. Rather than changing `u_mousedowninside`, a `u_mouseover` field is set when the

mouseenter message is received, and cleared when themouseleave method is received. And again, after this variable is manipulated, we repaint the box with `jbox_redraw()`.

```
void uisimp_mouseenter(t_uisimp **x, t_object *patcherview, t_pt pt, long modifiers)
{
    x->u_mouseover = true;
    jbox_redraw((t_jbox *)x);
}
void uisimp_mouseleave(t_uisimp **x, t_object *patcherview, t_pt pt, long modifiers)
{
    x->u_mouseover = false;
    jbox_redraw((t_jbox *)x);
}
```

## 11.10 Freeing a UI Object

If your object has created any clocks or otherwise allocated memory that should be freed when the object goes away, you should handle this in the free routine. But, most importantly, you must call the function `jbox_free()`. If your UI object doesn't need to do anything special in its free routine, you can pass `jbox_free()` as the free routine argument to `class_new()` in your initialization routine. We chose not to do this, since having an actual function permits easy modification should some memory need to be freed at some point in the future evolution of the object.

```
void uisimp_free(t_uisimp **x)
{
    jbox_free((t_jbox *)x);
}
```



## Chapter 12

# File Handling

Max contains a cross-platform set of routines for handling files.

These routines permit you to search for files, show file open and save dialogs, as well as open, read, write, and close them. The file API is based around a "path identifier" – a number that describes the location of a file. When searching or reading a file, path identifiers can be either a folders or collectives. Path identifiers that are negative (or zero) describe actual folders in the computer's file system, while path identifiers that are positive refer to collectives.

A basic thing you might want to do make your object accept the read message in a manner similar to existing Max objects. If the word read is followed by no arguments, a file dialog appears for the user to choose a file. If read is followed by an argument, your object will search for the file. If a file is found (or chosen), your object will open it and read data from it.

First, make your object accept the read message. The simplest way to make the filename argument optional is to use the [A\\_DEFSYM](#) argument type specifier. When the symbol argument is not present, Max passes your method the empty symbol.

```
class_addmethod(c, (method)myobject_read, "read", A_DEFSYM, 0);
```

The next requirement for any method that reads files is that it must defer execution to the low-priority thread, as shown in the following implementation, where the filename argument is passed as the symbol argument to defer.

```
void myobject_read(t_myobject **x, t_symbol *s)
{
    defer(x, (method)myobject_doread, s, 0, NULL);
}
```

The myobject\_doread() function compares the filename argument with the empty symbol – if the argument was not supplied, the [open\\_dialog\(\)](#) is used, otherwise, we call [locatefile\\_extended\(\)](#) to search for the file. This object looks for text files, so we use a four-character code 'TEXT' as our file type to either open or locate. File type codes define a set of acceptable extensions. The file max-fileformats.txt permits contains standard definitions, and you can add your own by creating a similar text file and placing it in the init folder inside the Cycling '74 folder.

```
void myobject_doread(t_myobject **x, t_symbol *s)
{
    t_fourcc filetype = 'TEXT', outtype;
    short numtypes = 1;
    char filename[MAX_PATH_CHARS];
    short path;
    if (s == gensym("")) { // if no argument supplied, ask for file
        if (open_dialog(filename, &path, &outtype, &filetype, 1)) // non-zero: user cancelled
            return;
    } else {
        strcpy(filename, s->s_name); // must copy symbol before calling locatefile_extended
        if (locatefile_extended(filename, &path, &outtype, &filetype, 1)) { // non-zero: not found
            object_error(x, "%s: not found", s->s_name);
            return;
    }
}
```

```

        }
    // we have a file
    myobject_openfile(x, filename, path);
}

```

To open and read files, you can use the cross-platform sysfile API. Files can be opened using a filename plus path identifier. If successfully opened, the file can be accessed using a [t\\_filehandle](#). Note that "files" inside collective files are treated identically to regular files, with the exception that they are read-only.

## 12.1 Reading Text Files

First, we'll implement reading the text file whose name and path identifier are passed to `myobject_openfile()` using a high-level routine [sysfile\\_readtextfile\(\)](#) specifically for reading text files that handles text encoding conversion for you. If you are reading text files, using this routine is strongly recommended since converting text encodings is unpleasant to say the least.

```

void myobject_openfile(t_myobject **x, char *filename, short path)
{
    t_filehandle fh;
    char **texthandle;
    if (path_opensysfile(filename, path, &fh, READ_PERM)) {
        object_error(x, "error opening %s", filename);
        return;
    }
    // allocate some empty memory to receive text
    texthandle = sysmem_newhandle(0);
    sysfile_readtextfile(fh, texthandle, 0, 0);      // see flags explanation below
    post("the file has %ld characters", sysmem_handlesize(texthandle));
    sysfile_close(fh);
    sysmem_freehandle(texthandle);
}

```

In most situations, you will pass 0 for the final two arguments to [sysfile\\_readtextfile\(\)](#). The third argument specifies a maximum length to read, but if the value is 0, the entire file is read in, regardless of its size. The final argument is a set of flags specifying options for reading in the text. The options concern the conversion of line breaks, text encoding, and the ability to add a null character to the end of the data returned.

Line breaks are converted on the basis of any line break flags. When reading text files, Max converts line breaks to "native" format, which is  
`\r\n`

on Windows and  
`\n`

on the Mac; this is the behavior you get if you either pass no line break flags or use [TEXT\\_LB\\_NATIVE](#). Other options include [TEXT\\_LB\\_MAC](#), [TEXT\\_LB\\_UNIX](#), or [TEXT\\_LB\\_PC](#).

By default, text files are converted from their source encoding to UTF-8. If you do not want this conversion to occur, you can use the [TEXT\\_ENCODING\\_USE\\_FILE](#) flag. This puts the burden on determining the encoding on you, which is probably not what you want. For example, the source text file might use UTF-16 encoding, which requires very different parsing than an 8-bit encoding.

Finally, you can have the memory returned from [sysfile\\_readtextfile\(\)](#) terminated with a NULL character if you use the [TEXT\\_NULL\\_TERMINATE](#) flag.

## 12.2 Reading Data Files

To read data files where you do not want to do text encoding conversion or worry about line breaks, you can use the same technique shown above for text files, but write the myobject\_openfile function using `sysfile_read()` instead of `sysfile_readtextfile()`. This example shows how to read an entire file into a single block of memory.

```
void myobject_openfile(t_myobject **x, char *filename, short path)
{
    t_filehandle fh;
    char *buffer;
    long size;
    if (path_opensysfile(filename, path, &fh, READ_PERM)) {
        object_error(x, "error opening %s", filename);
        return;
    }
    // allocate memory block that is the size of the file
    sysfile_geteof(fh, &size);
    buffer = sysmem_newptr(size);
    // read in the file
    sysfile_read(fh, &size, buffer);
    sysfile_close(fh);
    // do something with data in buffer here
    sysmem_freeptr(buffer);           // must free allocated memory
}
```

## 12.3 Writing Files

Some Max objects respond to the write message to save data into a file. If there is no argument present after the word write, a save file dialog is shown and the user specifies a file name and location. If an argument is present, it can either specify a complete path name or a filename. In the filename case, the file is written to the current "default" directory, which is the location where a patcher was last opened. In the full pathname case, the file is written to the location specified by the pathname.

Here's how to implement this behavior. We'll show how to handle the message arguments, then provide text and data file writing examples.

Message and argument handling is very similar to the way we implemented the read message above, including the use of deferred execution.

```
class_addmethod(c, (method)myobject_write, "write", A_DEFSYM, 0);
void myobject_write(t_myobject **x, t_symbol *s)
{
    defer(x, (method)myobject_dowrite, s, 0, NULL);
}
```

The myobject\_dowrite() function compares the filename argument with the empty symbol – if the argument was not supplied, `saveasdialog_extended()` is used to obtain the user's choice for filename and location. Our first example looks for text files, so we use a four-character code 'TEXT' as our file type for saving. File type codes define a set of acceptable extensions. The file max-fileformats.txt permits contains standard definitions, and you can add your own by creating a similar text file and placing it in the init folder inside the Cycling '74 folder.

```
void myobject_dowrite(t_myobject **x, t_symbol *s)
{
    long filetype = 'TEXT', outtype;
    short numtypes = 1;
    char filename[512];
    short path;
    if (s == gensym("")) {           // if no argument supplied, ask for file
        if (saveasdialog_extended(filename, &path, &outtype, &filetype, 1))           // non-zero: user cancelled
            return;
    } else {
        strcpy(filename, s->s_name);
        path = path_getdefault();
    }
    myobject_writefile(x, filename, path);
}
```

Here is the text file variant of myobject\_writefile() using the high-level `sysfile_writetextfile()` routine. We just write a sentence as our "text file" but your object will presumably have some text data stored internally that it will write. The buffer passed to `sysfile_writetextfile()` must be NULL-terminated, and will be assumed to be UTF-8 encoded.

Note that `path_createsysfile()` can accept a full path in the filename argument, in which case, the path argument is ignored. This means your object's write message can either accept a filename or full pathname and you needn't do anything special to accept both.

```
void myobject_writefile(t_myobject *x, char *filename, short path)
{
    char *buf = "write me into a file";
    long err;
    t_filehandle fh;
    err = path_createsysfile(filename, path, 'TEXT', &fh);
    if (err)
        return;
    err = sysfile_writetextfile(fh, &buf, TEXT_LB_NATIVE);
    sysfile_close(fh);
}
```

Here is a data file variant of myobject\_writefile(). It writes a small buffer of ten numbers to a file.

```
void myobject_writefile(t_myobject *x, char *filename, short path)
{
    char *buf[10];
    long count, i;
    long err;
    t_filehandle fh;
    // create some data
    for (i = 0; i < 10; i++)
        buf[i] = i + 1;
    count = 10;
    err = path_createsysfile(filename, path, 'TEXT', &fh);
    if (err)
        return;
    err = sysfile_write(fh, &count, buf);
    sysfile_close(fh);
}
```

# Chapter 13

## Scripting the Patcher

Your object can use scripting capabilities of the patcher to learn things about its context, such as the patcher's name, hierarchy, or the peer objects to your object in its patcher.

You can also modify a patcher, although any actions your object takes are not undoable and may not work in the runtime version.

### 13.1 Knowing the Patcher

To obtain the patcher object containing your object, you can use the obex hash table. The obex (for "object extensions") is, more generally, a way to store and recall data in your object. In this case, however, we are just using it in a read-only fashion.

Note that unlike the technique discussed in previous versions of the SDK, using the obex to find the patcher works at any time, not just in the new instance routine.

```
void myobject_getmypatcher(t_myobject *x)
{
    t_object *mypatcher;
    object_obex_lookup(x, gensym("#P"), &mypatcher);
    post("my patcher is at address %lx", mypatcher);
}
```

The patcher is an opaque Max object. To access data in a patcher, you'll use attributes and methods.

#### 13.1.1 Patcher Name and File Path

To obtain the name of the patcher and its file path (if any), obtain attribute values as shown below.

```
t_symbol *name = object_attr_getsym(patcher, gensym("name"));
t_symbol *path = object_attr_getsym(patcher, gensym("filepath"));
```

These attributes may return NULL or empty symbols.

### 13.1.2 Patcher Hierarchy

To determine the patcher hierarchy above the patcher containing your object, you can use `jpatcher_get_parentpatcher()`. A patcher whose parent is NULL is a top-level patcher. Here is a loop that prints the name of each parent patcher as you ascend the hierarchy.

```
t_object *parent, *patcher;
t_symbol *name;
object_obex_lookup(x, gensym("#P"), &patcher);
parent = patcher;
do {
    parent = jpatcher_get_parentpatcher(parent);
    if (parent) {
        name = object_attr_getsym(parent, gensym("name"));
        if (name)
            post("%s", name->s_name)
    }
} while (parent != NULL);
```

### 13.1.3 Getting Objects in a Patcher

To obtain the first object in a patcher, you can use `jpatcher_get_firstobject()`. Subsequent objects are available with `jbox_get_nextobject()`.

If you haven't read the [Anatomy of a UI Object](#), we'll mention that the patcher does not keep a list of non-UI objects directly. Instead it keeps a list of UI objects called boxes, and the box that holds non-UI objects is called a newobj. The "objects" you obtain with calls such as `jpatcher_get_firstobject()` are boxes. The `jbox_get_object()` routine can be used to get the pointer to the actual object, whether the box is a UI object or a newobj containing a non-UI object. In the case of UI objects such as dials and sliders, the pointer returned by `jbox_get_object()` will be the same as the box. But for non-UI objects, it will be different.

Here is a function that prints the class of every object (in a box) in a patcher containing an object.

```
void myobject_printpeers(t_myobject **x)
{
    t_object *patcher, *box, *obj;
    object_obex_lookup(x, gensym("#P"), &patcher);
    for (box = jpatcher_get_firstobject(patcher); box; box = jbox_get_nextobject(box)) {
        obj = jbox_get_object(box);
        if (obj)
            post("%s", object_classname(obj)->s_name);
        else
            post("box with NULL object");
    }
}
```

### 13.1.4 Iteration Using Callbacks

As an alternative to the technique shown above, you can write a callback function for use with the patcher's iteration service. The advantage of using iteration is that you can descend into the patcher hierarchy without needing to know the details of the various objects that may contain subpatchers (patcher, poly~, bpatcher, etc.). If you want to iterate only at one level of a patcher hierarchy, you can do that too.

Your iteration function is defined as follows. It will be called on every box in a patcher (and, if you specify, the patcher's subpatchers).

```
long myobject_iterator(t_myobject **x, t_object *b);
```

The function returns 0 if iteration should continue, or 1 if it should stop. This permits you to use an iterator as a way to search for a specific object.

Here is an example of using an iterator function:

```
t_object *patcher;
long result = 0;
t_max_err err;
err = object_obex_lookup(x, gensym("#P"), &patcher);
object_method(patcher, gensym("iterate"), myobject_iterator, (void *)x, PI_WANTBOX | PI_DEEP, &result);
```

The `PI_WANTBOX` flag tells the patcher iterator that it should pass your iterator function the box, rather than the object contained in the box. The `PI_DEEP` flag means that the iteration will descend, depth first, into subpatchers. The result parameter returns the last value returned by the iterator. For example, if the iterator terminates early by returning a non-zero value, it will contain that value. If the iterator function does not terminate early, result will be 0.

Assuming the iterator function receives boxes, here is an example iterator that prints out the class and scripting name (if any) of all of the objects in a patcher. Note that the scripting name is an attribute of the box, while the class we would like to know is of the object associated with the box.

```
long myobject_iterator(t_myobject **, t_object *b)
{
    t_symbol *name = object_attr_getsym(b, gensym("varname"));
    t_symbol *cls = object_classname(jbox_get_object(b));
    if (name)
        post("%s (%s)", cls->s_name, name->s_name);
    else
        post("%s", cls->s_name);
    return 0;
}
```

## 13.2 Creating Objects

Much of the Max user interface is implemented using patcher scripting. For example, the inspectors are patchers in which an inspector object has been created. The file browser window has four or five separate scripted objects in it. Even the debug window is a dynamically scripted patcher. We point this out just to inform you that creating objects in a patcher actually works (if you get all the details right). The xxx example object shows how to use patcher scripting to create an "editing window" similar to the ones you see when double-clicking on a table or buffer~ object.

Creating objects in a patcher generally requires the use of a [Dictionary](#) (see discussion of UI objects above), but there is a convenience function `newobject_sprintf()` that can be used to avoid some of the complexity.

To create an object, your task is to set some attributes. In the absence of any specific values, an object's attributes will be set to some default, but you'll probably care, at the very least, about specifying the object's location. Here is an example that creates a toggle and metro object using a combination of attribute parse syntax and sprintf. If you're interested in creating objects with `newobject_sprintf()`, it may help to examine a Max document to see some of the attribute name - value pairs used to specify objects.

```
t_object *patcher, *toggle, *metro;
t_max_err err;
err = object_obex_lookup(x, gensym("#P"), &patcher);
toggle = newobject_sprintf(patcher, "@maxclass toggle
@patching_position %2f %2f",
                          x->togxpos, x->togxpos);
metro = newobject_sprintf(patcher, "@maxclass newobj @text \"metro 400\
@patching_position %2f %2f\",
                           x->metxpos, x->metypos);
```

Note that to create a non-UI object, you use set the maxclass attribute to newobj and the text attribute to the contents of the object box. Attributes can be specified in any order. Using the patching\_position attribute permits you to specify only the top-left corner and use the object's default size. For text objects, the default size is based on the default font for the patcher.

Finally, note that `newobject_sprintf()` returns a pointer to the newly created box, not the newly created object inside the box. To get the object inside the box, use `jbox_get_object()`.

### 13.2.1 Connecting Objects

If you'd like to script the connections between two objects, you can do so via a message to the patcher. Assuming you have the patcher, toggle, and metro objects above, you'll create an array of atoms to send the message using `object_method_typed()`.

```
t_atom msg[4], rv;
atom_setobj(msg, toggle);           // source
atom_setlong(msg + 1, 0);          // outlet number (0 is leftmost)
atom_setobj(msg + 2, metro);        // destination
atom_setlong(msg + 3, 0);          // inlet number (0 is leftmost)
object_method_typed(patch, gensym("connect"), 4, msg, &rv);
```

If you want to have a hidden connection, pass an optional fifth argument that is any negative number.

## 13.3 Deleting Objects

To delete an object in a patcher you call `object_free()` on the box. As of Max 5.0.6 this will properly redraw the patcher and remove any connected patch cords.

## 13.4 Obtaining and Changing Patcher and Object Attributes

You can use object attribute functions to modify the appearance and behavior of objects in a patcher or the patcher itself. Note that only a few of these attributes can be modified by the user. The C level access to attributes is much more extensive.

Attributes whose type is object can be accessed via `object_attr_getobj()` / `object_attr_setobj()`. Attributes whose type is char can be accessed with `object_attr_getchar()` / `object_attr_setchar()`. Attributes whose type is long can be accessed with `object_attr_getlong()` / `object_attr_setlong()`. Attributes whose type is symbol can be accessed via `object_attr_getsym()` / `object_attr_setsym()`. For attributes that are arrays, such as colors and rectangles, use `object_attr_getvalueof()` / `object_attr_setvalueof()`.

### 13.4.1 Patcher Attributes

Name	Type	Settable	Description
box	object	No	The box containing the patcher (NULL for top-level patcher)
locked	char	Yes (not in runtime)	Locked state of the patcher
presentation	char	Yes	Presentation mode of the patcher
openinpresentation	char	Yes	Will patcher open in presentation mode?
count	long	No	Number of objects in a patcher
fccount	long	No	Number of objects in the patcher's foreground layer
bgcount	long	No	Number of objects in the patcher's background layer
numviews	long	No	Number of currently open views of the patcher
numwindowviews	long	No	Number of currently open window-based views of the patcher
firstobject	object	No	First box in the patcher
lastobject	object	No	Last box in the patcher
firstline	object	No	First patch cord in the patcher

firstview	object	No	First view object in the patcher
title	symbol	Yes	Window title
fulltitle	symbol	No	Complete title including "unlocked" etc.
name	symbol	No	Name (could be different from title)
filename	symbol	No	Filename
filepath	symbol	No	File path (platform-independent file path syntax)
fileversion	long	No	File version
noedit	char	No	Whether patcher can be unlocked
collective	object	No	Collective object, if patcher is inside a collective
cansave	char	No	Whether patcher can be saved
dirty	char	Yes (not in runtime)	Whether patcher is modified
bglocked	char	Yes	Whether background is locked
rect	double[4]	Yes	Patcher's rect (left, top, width, height)
defrect	double[4]	Yes	Patcher's default rect (used when opening the first view)
openrect	double[4]	Yes	Fixed initial window location
parentpatcher	object	No	Immediate parent patcher (NULL for toplevel patchers)
toppatcher	object	No	Topmost parent patcher (NULL for toplevel patchers)
parentclass	object	No	Class object of parent (patcher, poly~, bpatcher etc.)
bgcolor	double[4]	Yes	Locked background color (RGBA)
editing_bgcolor	double[4]	Yes	Unlocked background color (RGBA)
edit_framecolor	double[4]	Yes	Text editing frame color
locked_iocolor	double[4]	Yes	Locked inlet/outlet color
unlocked_iocolor	double[4]	Yes	Unlocked inlet/outlet color
boguscolor	double[4]	Yes	Color of uninitialized (bogus) objects
gridsize	double[2]	Yes	Editing grid size
gridonopen	char	Yes	Show grid on open
gridsnapopen	char	Yes	Snap to grid on open
imprint	char	Yes	Save default-valued object attributes
defaultfocusbox	symbol	Yes	Default focus box (varname)
enablehscroll	char	Yes	Show horizontal scrollbar
enablevscroll	char	Yes	Show vertical scrollbar
boxanimatetime	long	Yes	Box animation time
default_fontname	symbol	Yes	Default font name
default_fontface	long	Yes	Default "fake" font face (0 plain, 1 bold, 2 italic, 3 bold italic)
default_fontsize	long	Yes	Default font size in points
toolbarvisible	char	Yes	Show toolbar on open
toolbarheight	long	Yes	Height of toolbar (can use 0 for invisible)
toolbarid	symbol	Yes	Name (in maxinterface.json) of toolbar, none = empty symbol

### 13.4.2 Box Attributes

Name	Type	Settable	Description
rect	double[4]	Settable only	Changes both patching_rect and presentation_rect
presentation_rect	double[4]	Yes	Presentation mode rect
patching_rect	double[4]	Yes	Patching mode rect

position	double[2]	Settable only	Changes both patching_position and presentation_position
size	double[2]	Settable only	Changes both patching_size and presentation_size
patching_position	double[2]	Yes	Patching mode position (top, left corner)
presentation_position	d[2]	Yes	Presentation mode position
patching_size	double[2]	Yes	Patching mode size (width, height)
presentation_size	double[2]	Yes	Presentation mode size
maxclass	symbol	No	Name of Max class (newobj for non-UI objects)
object	object	No	Associated object (equivalent to jbox_get_object)
patcher	object	No	Containing patcher
hidden	char	Yes	Is box hidden on lock?
fontname	symbol	Yes	Font name (if box has font attributes or a text field)
fontface	long	Yes	"Fake" font face (if box has font attribute or a text field)
fontsize	long	Yes	Font size (if box has font attributes or a text field)
textcolor	double[4]	Yes	Text color (if box has font attributes or a text field)
hint	symbol	Yes	Associated hint
color	double[4]	Yes	Standard color attribute (may not be present in all objects)
nextobject	object	No	Next object in the patcher's list
prevobject	object	No	Previous object in the patcher's list
varname	symbol	Yes	Scripting name
id	symbol	No	Immutable object ID (stored in files)
canhilite	char	No	Does this object accept focus?
background	char	Yes	Include in background
ignoreclick	char	Yes	Ignores clicks
maxfilename	symbol	No	Filename if class is external
description	symbol	No	Description used by assistance
drawfirstin	char	No	Is leftmost inlet drawn?
growy	char	No	Can object grow with fixed aspect ratio?
growboth	char	No	Can object grow independently in width and height?
nogrow	char	No	Is object fixed size?
mousedragdelta	char	No	Does object use hidden-mouse drag tracking (number box)
textfield	object	No	Textfield object associated with this box if any
editactive	char	No	Is object the currently focused box in an unlocked patcher?
prototypename	symbol	No	Name of the prototype file used to create this object
presentation	char	Yes	Is object included in the presentation?
annotation	symbol	Yes	Text shown in clue window when mouse is over the object
numinlets	long	No	Number of inlets visible
numoutlets	long	No	Number of outlets visible
outlettype	symbol[]	No	Array of symbols with outlet types ("signal" etc.)

To access an attribute of a non-UI object, use [jbox\\_get\\_object\(\)](#) on the box to obtain the non-UI object first.

## Chapter 14

# Enhancements to Objects

### 14.1 Preset Support

Presets are a simple state-saving mechanism. Your object receives a preset message when state is being saved. You respond by creating a message that will be sent back to your object when the preset is recalled.

For more powerful and general state-saving, use the pattr system described below.

To support saving a single integer in a preset, you can use the `preset_int()` convenience function. The `preset_int()` function records an int message with the value you pass it in the preset, to be sent back to your object at a later time.

```
class_addmethod(c, (method)myobject_preset, "preset", 0);
void myobject_preset(t_myobject *x)
{
    preset_int(x, x->m_currentvalue);
}
```

More generally, you can use `preset_store()`. Here is an example of storing two values (`m_xvalue` and `m_yvalue`) in a list.

```
preset_store("ossll", x, ob_sym(x), gensym("list"), x->m_xvalue, x->m_yvalue);
```

### 14.2 Pattr Support

In most cases, you need only to define your object's state using [Attributes](#) and it will be ready for use with Max's pattr system. For more complex scenarios you may also wish to investigate [object\\_notify\(\)](#), [object\\_attach\(\)](#), and the section on [Receiving Notifications](#).

### 14.3 Assistance

To show descriptions of your object's inlets and outlets while editing a patcher, your object can respond to the assist message with a function that copies the text to a string.

```
class_addmethod(c, (method)myobject_assist, "assist", A_CANT, 0);
```

The function below has two inlets and one outlet. The io argument will be 1 for inlets, 2 for outlets. The index argument will be 0 for the leftmost inlet or outlet. You can copy a maximum of 512 characters to the output string s. You can use

`strncpy_zero()` to copy the string, or if you want to format the assistance string based on a current value in the object, you could use `snprintf_zero()`.

```
void myobject_assist(t_myobject *x, void *b, long io, long index, char *s)
{
    switch (io) {
        case 1:
            switch (index) {
                case 0:
                    strncpy_zero(s, "This is a description of the leftmost inlet", 512);
                    break;
                case 1:
                    strncpy_zero(s, "This is a description of the rightmost inlet", 512);
                    break;
            }
            break;
        case 2:
            strncpy_zero(s, "This is a description of the outlet", 512);
            break;
    }
}
```

## 14.4 Hot and Cold Inlets

Objects such as operators (+, -, etc.) and the int object have inlets that merely store values rather than performing an operation and producing output. These inlets are labeled with a blue color to indicate they are "cold" rather than action-producing "hot" inlets. To implement this labeling, your object can respond to the `inletinfo` message.

```
class_addmethod(c, (method)myobject_inletinfo, "inletinfo", A_CANT, 0);
```

If all of your object's non-left inlets are "cold" you can use the function `stdinletinfo()` instead of writing your own, as shown below:

```
class_addmethod(c, (method)stdinletinfo, "inletinfo", A_CANT, 0);
```

To write your own function, just look at the `index` argument (which is 0 for the left inlet). This example turns the third inlet cold. You don't need to do anything for "hot" inlets.

```
void myobject_inletinfo(t_myobject *x, void *b, long index, char *t)
{
    if (index == 2)
        *t = 1;
}
```

## 14.5 Showing a Text Editor

Objects such as coll and text display a text editor window when you double-click. Users can edit the contents of the objects and save the updated data (or not). Here's how to do the same thing in your object.

First, if you want to support double-clicking on a non-UI object, you can respond to the `dblclick` message.

```
class_addmethod(c, (method)myobject_dblclick, "dblclick", A_CANT, 0);
void myobject_dblclick(t_myobject *x)
{
    // open editor here
}
```

You'll need to add a `t_object` pointer to your object's data structure to hold the editor.

```
typedef struct _myobject
{
    t_object m_obj;
    t_object *m_editor;
} t_myobject;
```

Initialize the `m_editor` field to NULL in your new instance routine. Then implement the `dblclick` method as follows:

```
if (!x->m_editor)
```

```
x->m_editor = object_new(CLASS_NOBOX, gensym("jed"), (t_object *)x, 0);
else
    object_attr_setchar(x->m_editor, gensym("visible"), 1);
```

The code above does the following: If the editor does not exist, we create one by making a "jed" object and passing our object as an argument. This permits the editor to tell our object when the window is closed.

If the editor does exist, we set its visible attribute to 1, which brings the text editor window to the front.

To set the text of the edit window, we can send our jed object the settext message with a zero-terminated buffer of text. We also provide a symbol specifying how the text is encoded. For best results, the text should be encoded as UTF-8. Here is an example where we set a string to contain "Some text to edit" then pass it to the editor.

```
char text[512];
strcpy(text, "Some text to edit");
object_method(x->m_editor, gensym("settext"), text, gensym("utf-8"));
```

The title attribute sets the window title of the text editor.

```
object_attr_setsym(x->m_editor, gensym("title"), gensym("crazytext"));
```

When the user closes the text window, your object (or the object you passed as an argument when creating the editor) will be sent the edclose message.

```
class_addmethod(c, (method)myobject_edclose, "edclose", A_CANT, 0);
```

The edclose method is responsible for doing something with the text. It should also zero the reference to the editor stored in the object, because it will be freed. A pointer to the text pointer is passed, along with its size. The encoding of the text is always UTF-8.

```
void myobject_edclose(t_myobject *x, char **ht, long size)
{
    // do something with the text
    x->m_editor = NULL;
}
```

If your object will be showing the contents of a text file, you are still responsible for setting the initial text, but you can assign a file so that the editor will save the text data when the user chooses Save from the File menu. To assign a file, use the filename message, assuming you have a filename and path ID.

```
object_method(x->m_editor, gensym("filename"), x->m_myfilename, x->m_mypath);
```

The filename message will set the title of the text editor window, but you can use the title attribute to override the simple filename. For example, you might want the name of your object to precede the filename:

```
char titlename[512];
sprintf(titlename, "myobject: %s", x->m_myfilename);
object_attr_setsym(x->m_editor, gensym("title"), gensym(titlename));
```

Each time the user chooses Save, your object will receive an edsav message. If you return zero from your edsav method, the editor will proceed with saving the text in a file. If you return non-zero, the editor assumes you have taken care of saving the text. The general idea is that when the user wants to save the text, it is either updated inside your object, updated in a file, or both. As an example, the js object uses its edsav message to trigger a recompile of the Javascript code. But it also returns 0 from its edsav method so that the text editor will update the script file. Except for the return value, the prototype of the edsav method is identical to the edclose method.

```
class_addmethod(c, (method)myobject_edsav, "edsav", A_CANT, 0);
long myobject_edsav(t_myobject *x, char **ht, long size)
{
    // do something with the text
    return 0;           // tell editor it can save the text
}
```

## 14.6 Accessing Data in table Objects

Table objects can be given names as arguments. If a table object has a name, you can access the data using [table\\_get\(\)](#). Supply a symbol, as well as a place to assign a pointer to the data and the length. The following example accesses a table called foo, and, if found, posts all its values.

```
long **data = NULL;
long i, size;
if (!table_get(gensym("foo"), &data, &size)) {
    for (i = 0; i < size; i++) {
        post("%ld: %ld", i, (*data)[i]);
    }
}
```

You can also write data into the table. If you would like the table editor to redraw after doing so, use [table\\_dirty\(\)](#). Here's an example where we set all values in the table to zero, then notify the table to redraw.

```
long **data = NULL;
long i, size;
if (!table_get(gensym("foo"), &data, &size)) {
    for (i = 0; i < size; i++) {
        (*data)[i] = 0;
    }
    table_dirty(gensym("foo"));
}
```

# Chapter 15

## Data Structures

### 15.1 Available Data Structures

The Max API provides a variety of useful data structures which may be used across platforms and provide basic thread-safety.

- [Atom Array](#) : container for an array of atoms
- [Linked List](#) : doubly-linked-list
- [Hash Table](#) : hash table for mapping symbols to data
- [Quick Map](#) : a double hash with keys mapped to values and vice-versa
- [Database](#) : SQLite database access
- [Index Map](#) : managed array of pointers
- [String Object](#) : wrapper for C-strings with an API for manipulating them
- [Symbol Object](#) : wrapper for symbols
- [Dictionary](#) : structured/hierarchical data that is both sortable and fast

### 15.2 Passing Data Structures

Most often, the use of a particular instance of a data structure will be limited to within the confines a single class or object you create. However, in some cases you may wish to pass structured data from one object to another. For this purpose, Max 6 introduced facilities for passing named [`t\_dictionary`](#) instances.

Examples, descriptions, and API documentation can be found in [Dictionary Passing API](#).



# Chapter 16

## Threading

The Max systhread API has two main purposes.

First, it can be used to implement thread protection which works in conjunction with Max's existing threading model and is cross-platform. Thread protection prevents data corruption in the case of simultaneously executing threads in the same application. We'll discuss the Max threading model and show you a simple example of thread protection below, but you can often avoid the need to use thread protection by using one of the thread-safe [Data Storage](#) Max provides.

The second use of the systhread API is a cross-platform way to create and manage threads. This is an advanced feature that very few programmers will ever need. For information on creating and managing threads look at the systhread API header file.

### 16.1 Max Threading Operation

Please note that this description of how Max operates is subject to change and may not apply to future versions. For more information about the Max scheduler and low-priority queue, see the [The Scheduler](#) section.

Max (without audio) has two threads. The main or event thread handles user interaction, asks the system to redraw the screen, processes events in the low-priority queue. When not in Overdrive mode, the main thread handles the execution of events in the Max scheduler as well. When Overdrive is enabled, the scheduler is moved to a high-priority timer thread that, within performance limits imposed by the operating system, attempts to run at the precise scheduler interval set by user preference. This is usually 1 or 2 milliseconds.

The basic idea is to put actions that require precise timing and are relatively computationally cheap in the high-priority thread and computationally expensive events that do not require precise timing in the main thread. On multi-core machines, the high-priority thread may (or may not) be executing on a different core.

On both Mac and Windows, either the main thread or the timer thread can interrupt the other thread, even though the system priority level of the timer thread is generally much higher. This might seem less than optimal, but it is just how operating systems work. For example, if the OS comes to believe the Max timer thread is taking too much time, the OS may "punish" the thread by interrupting it with other threads, even if those threads have a lower system priority.

Because either thread can be interrupted by the other, it is necessary to use thread protection to preserve the integrity of certain types of data structures and logical operations. A good example is a linked list, which can be corrupted if a thread in the process of modifying the list is interrupted by another thread that tries to modify the list. The Max [t\\_linklist](#) data

structure is designed to be thread-safe, so if you need such a data structure, we suggest you use [t\\_linklist](#). In addition, Max provides thread protection between the timer thread and the main thread for many of its common operations, such as sending messages and using outlets.

When we add audio into the mix (so to speak), the threading picture gets more complicated. The audio perform routine is run inside a thread that is controlled by the audio hardware driver. In order to eliminate excessive thread blocking and potential race conditions, the thread protection offered inside the audio perform routine is far less comprehensive, and as discussed in the MSP section of the API documentation, the only supported operation for perform routines to communicate to Max is to use a clock. This will trigger a function to run inside the Max scheduler.

The Max scheduler can be run in many different threading conditions. As explained above it can be run either in the main thread or the timer thread. When Scheduler in Audio Interrupt (SIAI) is enabled, the scheduler runs with an interval equal to every signal vector of audio inside the audio thread. However, if the Non-Real-Time audio driver is used, the audio thread is run inside the main thread, and if SIAI is enabled, the scheduler will also run inside the main thread. If not, it will run either in the main thread or the timer thread depending on the Overdrive setting. (Using the Non-Real-Time audio driver without SIAI will generally lead to unpredictable results and is not recommended.)

## 16.2 Thread Protection

The easiest method for thread protection is to use critical sections. A critical section represents a region of code that cannot be interrupted by another thread. We speak of entering and exiting a critical section, and use [critical\\_enter\(\)](#) and [critical\\_exit\(\)](#) to do so.

Max provides a default global critical section for your use. This same critical section is used to protect the timer thread from the main thread (and vice versa) for many common Max data structures such as outlets. If you call [critical\\_enter\(\)](#) and [critical\\_exit\(\)](#) with argument of 0, you are using this global critical section. Typically it is more efficient to use fewer critical sections, so for many uses, the global critical section is sufficient. Note that the critical section is recursive, so you if you exit the critical section from within some code that is already protected, you won't be causing any trouble.

### 16.2.1 When Messages Arrive

It's possible that a message sent to your object could interrupt the same message sent to your object ("myobject"). For example, consider what happens when a button is connected to the left inlet of myobject and a metro connected to the same inlet.

When a user clicks on the bang button, the message is sent to your object in the main thread. When Overdrive is enabled, the metro will send a bang message to your object in the timer thread. Either could interrupt the other. If your object performs operations on a data structure that cannot be interrupted, you should use thread protection.

### 16.2.2 Critical Section Example

Here is an example that uses the global critical section to provide thread protection for an array data structure. Assume we have an operation `array_read()` that reads data from an array, and `array_insert()` that inserts data into the same array. We wish to ensure that reading doesn't interrupt writing and vice versa.

```
long array_read(t_myobject *x, long index)
{
    critical_enter(0);
    result = x->m_data[index];
    critical_exit(0);
    return result;
```

```
}
```

Note that all paths of your code must exit the critical region once it is entered, or the other threads in Max will never execute.

```
long array_insert(t_myobject *x, long index, long value)
{
    critical_enter(0);
    // move existing data
    sysmem_copyptr(x->m_data + index, x->m_data + index + 1, (x->m_size - x->m_index) * sizeof(long));
    // write new data
    x->m_data[index] = value;
    critical_exit(0);
}
```



# Chapter 17

## Drag'n'Drop

The Max file browser permits you to drag files to a patcher window or onto objects to perform file operations.

Your object can specify the file types accepted as well as a message that will be sent when the user releases the mouse button with the file on top of the object. UI and non-UI objects use the same interface to drag'n'drop.

Example UI object: *pictmeter~*. Example non-UI: TBD.

Messages to support:

`acceptsdrag_locked` ([A\\_CANT](#))

Sent to an object during a drag when the mouse is over the object in an unlocked patcher.

`acceptsdrag_unlocked` ([A\\_CANT](#))

Sent to an object during a drag when the mouse is over the object in a locked patcher.

### 17.1 Discussion

Why two different scenarios? `acceptsdrag_unlocked()` can be thought of as an "editing" operation. For example, objects such as *pictslider* accept new image files for changing their appearance when the patcher is unlocked, but not when the patcher is locked. By contrast, *sfplay~* can accept audio files for playback in either locked or unlocked patchers, since that is something you can do with a message (rather than an editing operation that changes the patcher).

Message handler definitions:

```
long myobject_acceptsdrag_unlocked(t_myobject **x, t_object *drag, t_object *view);  
long myobject_acceptsdrag_locked(t_myobject **x, t_object *drag, t_object *view);
```

The handlers return true if the file(s) contained in the drag can be used in some way by the object. To test the filetypes, use `jdrag_matchdragrole()` passing in the drag object and a symbol for the file type. Here is list of pre-defined file types:

- audiofile
- imagefile
- moviefile
- patcher

- helpfile
- textfile

or to accept all files, use file

If jdrag\_matchdragrole() returns true, you then describe the messages your object receives when the drag completes using jdrag\_object\_add(). You can add as many messages as you wish. If you are only adding a single message, use jdrag\_object\_add(). For more control over the process, and for adding more than one message, jdrag\_add() can be used. If you add more than one message, the user can use the option key to specify the desired action. By default, the first one you add is used. If there are two actions, the option key will cause the second one to be picked. If there are more than two, a pop-up menu appears with descriptions of the actions (as passed to jdrag\_add()), and the selected action is used.

Example:

This code shows how to respond to an audiofile being dropped on your object by having the read message sent.

```
if (jdrag_matchdragrole(drag, gensym("audiofile"), 0)) {
    jdrag_object_add(drag, (t_object *)x, gensym("read"));
    return true;
}
return false;
```

Your acceptsdrag handler can test for multiple types of files and add different messages.

# Chapter 18

## ITM

ITM is the tempo-based timing system introduced with Max 5.

It allows users to express time in tempo-relative units as well as milliseconds, samples, and an ISO 8601 hour-minute-second format. In addition, ITM supports one or more transports, which can be synchronized to external sources. An ITM-aware object can schedule events to occur when the transport reaches a specific time, or find out the current transport state.

The ITM API is provided on two different levels. The time object ([t\\_timeobject](#)) interface provides a higher-level way to parse time format information and schedule events. In addition, you can use lower-level routines to access ITM objects ([t\\_itm](#)) directly. An ITM object is responsible for maintaining the current time and scheduling events. There can be multiple ITM objects in Max, each running independently of the others.

### 18.1 Scheduling Temporary Events

There are two kinds of events in ITM. Temporary events are analogous to Max clock objects in that they are scheduled and fire at a dynamically assigned time. Once they have executed, they are removed from the scheduler. Permanent events always fire when the transport reaches a specific time, and are not removed from the scheduler. The ITM-aware metro is an example of an object that uses temporary events, while the timepoint object uses permanent events. We'll show how to work both types using an example included in the SDK called *delay2*. The existing Max delay object provides this capability, but this example shows most of the things you can do with the time object interface. To see the complete object, look at the *delay2 example*. We'll introduce a simpler version of the object, then proceed to add the quantization and the additional outlet that generates a delayed bang based on low-level ITM calls.

The ITM time object API is based on a Max object you create that packages up common ways you will be using ITM, including attribute support, quantization, and, if you want it, the ability to switch between traditional millisecond-based timing and tempo-based timing using an interface that is consistent with the existing Max objects such as metro and delay. (If you haven't familiarized yourself with attributes, you may want to read through the discussion about them in [Attributes](#) before reading further.)

To use the time object, you'll first need to provide some space in your object to hold a pointer to the object(s) you'll be creating.

```
typedef struct _delay2simple
{
    t_object m_obj;
    t_object *m_timeobj;
    void *m_outlet;
```

```
} _delay2simple;
```

Next, in your `ext_main()` routine, you'll create attributes associated with the time object using the `class_time_addattr()` function.

```
class_time_addattr(c, "delaytime", "Delay Time", TIME_FLAGS_TICKSONLY | TIME_FLAGS_USECLOCK |
    TIME_FLAGS_TRANSPORT);
```

The second argument, "delaytime", is a string that names the attribute. Users of your object will be able to change the delay value by sending a delaytime message. "Delay Time" is the label users see for the attribute in the inspector. The flags argument permits you to customize the type of time object you'd like. `TIME_FLAGS_TICKSONLY` means that the object can only be specified in tempo-relative units. You would not use this flag if you want the object to use the regular Max scheduler if the user specifies an absolute time (such as milliseconds). `TIME_FLAGS_USECLOCK` means that it is a time object that will actually schedule events. If you do not use this flag, you can use the time object to hold and convert time values, which you use to schedule events manually. `TIME_FLAGS_TRANSPORT` means that an additional attribute for specifying the transport name is added to your object automatically (it's called "transport" and has the label "Transport Name"). The combination of flags above is appropriate for an object that will be scheduling events on a temporary basis that are only synchronized with the transport and specified in tempo-relative units.

The next step is to create a time object in your new instance routine using `time_new`. The `time_new` function is something like `clock_new` – you pass it a task function that will be executed when the scheduler reaches a certain time (in this case, `delay2simple_tick`, which will send out a bang). The first argument to `time_new` is a pointer to your object, the second is the name of the attribute created via `class_time_addattr`, the third is your task function, and the fourth are flags to control the behavior of the time object, as explained above for `class_time_addattr`.

Finally, we use `time_setvalue` to set the initial delay value to 0.

```
void *_delay2simple_new()
{
    t_delay2simple *x;
    t_atom a;
    x = (t_delay2simple *)object_alloc(s_delay2simple_class);
    x->m_timeobj = (t_object *)time_new((t_object *)x, gensym("delaytime"), (method)delay2simple_tick,
        TIME_FLAGS_TICKSONLY | TIME_FLAGS_USECLOCK);
    x->m_outlet = bangout((t_object *)x);
    atom_setfloat(&a, 0.);
    time_setvalue(x->d_timeobj, NULL, 1, &a);
    return x;
}
```

To make a delayed bang, we need a `delay2simple_bang` function that causes our time object to put its task function into the ITM scheduler. This is accomplished using `time_schedule`. Note that unlike the roughly equivalent `clock_fdelay`, where the delay time is an argument, the time value must already be stored inside the time object using `time_setvalue`. The second argument to `time_schedule` is another time object that can be used to control quantization of an event. Since we aren't using quantization in this simple version of `delay2`, we pass `NULL`.

```
void delay2simple_bang(t_delay2 *x)
{
    time_schedule(x->d_timeobj, NULL);
}
```

Next, our simple task routine, `delay2simple_tick`. After the specified number of ticks in the time object has elapsed after the call to `time_schedule`, the task routine will be executed.

```
void delay2_tick(t_delay2 *x)
{
    outlet_bang(x->d_outlet);
}
```

Now let's add the two more advanced features found in `delay2`: quantization and a second (unquantized) bang output using low-level ITM routines. Here is the `delay2` data structure. The new elements are a proxy (for receiving a delay time), a time object for quantization (`d_quantize`), a clock to be used for low-level ITM scheduling, and an outlet for the use of the low-level clock's task.

```
typedef struct delay2
{
    t_object d_obj;
    void *d_outlet;
```

```

void *d_proxy;
long d_inletnum;
t_object *d_timeobj;
t_object *d_outlet2;
t_object *d_quantize;
void *d_clock;
} t_delay2;

```

In the initialization routine, we'll define a quantization time attribute to work in conjunction with the `d_quantize` time object we'll be creating. This attribute does not have its own clock to worry about. It just holds a time value, which we specify will only be in ticks (quantizing in milliseconds doesn't make sense in the ITM context). If you build `delay2` and open the inspector, you will see time attributes for both Delay Time and Quantization.

```
class_time_addattr(c, "quantize", "Quantization", TIME_FLAGS_TICKSONLY);
```

Here is part of the revised `delay2` new instance routine. It now creates two time objects, plus a regular clock object.

```

x->d_inletnum = 0;
x->d_proxy = proxy_new(x, 1, &x->d_inletnum);
x->d_outlet2 = bangout(x);
x->d_outlet = bangout(x);
x->d_timeobj = (t_object*) time_new((t_object *)x, gensym("delaytime"), (method)delay2_tick,
    TIME_FLAGS_TICKSONLY | TIME_FLAGS_USECLOCK);
x->d_quantize = (t_object*) time_new((t_object *)x, gensym("quantize"), NULL, TIME_FLAGS_TICKSONLY);
x->d_clock = clock_new((t_object *)x, (method)delay2_clocktick);

```

To use the quantization time object, we can pass it as the second argument to `time_schedule`. If the value of the quantization is 0, there is no effect. Otherwise, `time_schedule` will move the event time so it lies on a quantization boundary. For example, if the quantization value is 4n (480 ticks), the delay time is 8n (240 ticks) and current time is 650 ticks, the delay time will be adjusted so that the bang comes out of the `delay2` object at 980 ticks instead of 890 ticks.

In addition to using quantization with `time_schedule`, `delay2_bang` shows how to calculate a millisecond equivalent for an ITM time value using `itm_ticksoms`. This delay value is not quantized, although you read the time value from the `d_quantize` object and calculate your own quantized delay if wanted. The "calculated" delay is sent out the right outlet, since the clock we created uses `delay2_clocktick`.

```

void delay2_bang(t_delay2 **x)
{
    double ms, tix;
    time_schedule(x->d_timeobj, x->d_quantize);
    tix = time_getticks(x->d_timeobj);
    tix += (tix / 2);
    ms = itm_ticksoms(time_getitm(x->d_timeobj), tix);
    clock_fdelay(x->d_clock, ms);
}
void delay2_clocktick(t_delay2 **x)
{
    outlet_bang(x->d_outlet2);
}

```

## 18.2 Permanent Events

A permanent event in ITM is one that has been scheduled to occur when the transport reaches a specific time. You can schedule a permanent event in terms of ticks or bars/beats/units. An event based in ticks will occur when the transport reaches the specified tick value, and it will not be affected by changes in time signature. An event specified for a time in bars/beats/units will be affected by the time signature. As an example, consider an event scheduled for bar 2, beat 1, unit 0. If the time signature of the ITM object on which the event has been scheduled is 3/4, the event will occur at 480 times 3 or 1440 ticks. But if the time signature is 4/4, the event will occur at 1920 ticks. If, as an alternative, you had scheduled the event to occur at 1920 ticks, setting the time signature to 3/4 would not have affected when it occurred.

You don't "schedule" a permanent event. Once it is created, it is always in an ITM object's list of permanent events. To specify when the event should occur, use `time_setvalue`.

The high-level time object interface handles permanent events. Let's say we want to have a time value called "targettime." First, we declare an attribute using `class_time_addattr`. The flags used are `TIME_FLAGS_TICKSONLY`

(required because you can't specify a permanent event in milliseconds), `TIME_FLAGS_LOCATION` (which interprets the bar/beat/unit times where 1 1 0 is zero ticks), `TIME_FLAGS_PERMANENT` (for a permanent event), and `TIME_FLAGS_TRANSPORT` (which adds a transport attribute permitting a user to choose a transport object as a destination for the event) and `TIME_FLAGS_POSITIVE` (constrains the event to happen only for positive tick and bar/beat/unit values).

```
class_time_addattr(c, "targettime", "Target Time", TIME_FLAGS_TICKONLY | TIME_FLAGS_LOCATION |
    TIME_FLAGS_PERMANENT | TIME_FLAGS_TRANSPORT | TIME_FLAGS_POSITIVE);
```

The `TIME_FLAGS_TRANSPORT` flag is particularly nice. Without any intervention on your part, it creates a transport attribute for your object, and takes care of scheduling the permanent event on the transport the user specifies, with a default value of the global ITM object. If you want to cause your event to be rescheduled dynamically when the user changes the transport, your object can respond to the reschedule message as follows.

```
class_addmethod(c, (method)myobject_reschedule, "reschedule", A_CANT, 0); // for dynamic transport
    reassignment
```

All you need to do in your reschedule method is just act as if the user has changed the time value, and use the current time value to call `time_setvalue`.

In your new instance routine, creating a permanent event with `time_new` uses the same flags as were passed to `class_time_addattr`:

```
x->t_time = (t_object*) time_new((t_object *)x, gensym("targettime"), (method)myobject_tick,
    TIME_FLAGS_TICKONLY | TIME_FLAGS_USECLOCK | TIME_FLAGS_PERMANENT | TIME_FLAGS_LOCATION |
    TIME_FLAGS_POSITIVE);
```

The task called by the permanent time object is identical to a clock task or an ITM temporary event task.

## 18.3 Cleaning Up

With all time objects, both permanent and temporary, it's necessary to free the objects in your object's free method. Failure to do so will lead to crashes if your object is freed but its events remain in the ITM scheduler. For example, here is the `delay2` free routine:

```
void delay2_free(t_delay2 **x)
{
    freeobject(x->d_timeobj);
    freeobject(x->d_quantize);
    freeobject((t_object *) x->d_proxy);
    freeobject((t_object *)x->d_clock);
}
```

# Chapter 19

## MC

MC is the system for multi-voice or multi-channel audio signals introduced with Max 8.

MC is not an entirely new API for MSP objects. Instead it is built on top of the existing MSP API. Is it implemented as some additions to the MSP signal compiler — the code that turns a graph of MSP objects into an ordered sequence of operations on signals — to deal with patch cords that hold more than one audio signal. Multi-channel signal patch cords co-exist with regular old-school patch cords as well as Jitter and event patch cords.

An important principle of Max is that outlet types define patch cord types. This is how Max knows, when the user clicks on an outlet to patch something, what kind of patch cord to make. Jitter objects have “matrix” outlets. Regular Max events travel through outlets that either have no type defined or a type such as “int” or “bang.”

Following this principle, if you want your MSP object to have outlets that produce multi-channel signals, you will have to change the type of the outlets from signal to multichannelsignal. An outlet of type multichannelsignal will have between 1 and 1024 signals in it. A patch cord coming from this type of outlet will be a special color and width, and can be connected to any MSP object, even those that don’t know about MC. (In that case, only the first channel will be used.)

### 19.1 The MC Wrapper

What are you hoping to do to your object within the MC universe? If it’s a filter, do you want N filters, one operating on every audio channel? If that’s all you want, you can avoid doing any coding simply by whitelisting your object to use the MC wrapper. Let’s say your object is called myfilter~ and you want the N-way version to be called mc.myfilter~. Add the following message to Max in any file that is evaluated at startup (that typically means you’ll put it in the init folder):

```
max objectfile mc.myfilter~ mc.wrapper~ myfilter~;
```

This establishes a mapping when the user types mc.myfilter~ into an object box. The MC wrapper looks at the name the user types, removes the “mc.” from the beginning, and looks for a Max object with the string that remains. So, this means you can’t do this:

```
max objectfile myNWAYfilter~ mc.wrapper~ myfilter~;
```

The name “myfilter~” at the end of this message specifies the name of the help file to open. If you want to make a special help file for the N-way version of your object, you could do this:

```
max objectfile mc.myfilter~ mc.wrapper~ mc.myfilter~;
```

Finally, you may provide an optional fourth argument to the objectfile message that specifies the tab to open in the help patcher. For example, when you open the help patcher for the mcs.limi~ object in Max the limi~ help patcher is opened to the tab named “mcs”. This done using the pattern from this code:

```
max objectfile mc.myfilter~ mc.wrapper~ mc.myfilter~ <optional-helpfile-tabname>;
```

## 19.2 Multi-channel Inputs and Outputs

Maybe your concept of MC compatibility is not related to having N copies of your object in the wrapper. For example, your object might be concerned with audio signal input or output, either to the outside world or to a file. Perhaps you simply want to accept all inputs as a single multi-channel signal (which is nice if you don't want to decide in advance how many channels you will accept). Perhaps you want to produce a multi-channel signal instead of separate single-channel signals. Maybe your object mixed some stuff together that you realize would be nice not to mix together, so you'd like to provide each unmixed audio output together in a multi-channel patch cord.

For these cases, you can use the extensions to the MSP API described here. An MSP object that is MC-compatible will work in any version of Max with 64-bit floating-point. By convention, MSP objects that operate in both single-channel and multi-channel versions look at the object name symbol passed to the new instance routine and are multi-channel if the name begins with "mc." or "mcs." You are free to rebel against this standard and establish your own convention.

There are no special functions exported from Max or the MSP library specific to MC, so the only thing that will break in Max 7 and earlier versions is that your outlets won't work because only Max 8 knows about the multichannelsignal outlet type.

## 19.3 Creating Multi-channel Inlets and Outlets

For multi-channel inlets, you don't need to do anything special unless you want fewer inlets than you would otherwise. Just call `dsp_setup()` as you normally would and the user will be able to connect both single-channel and multi-channel patch cords. For multi-channel outlets, instead of

```
outlet_new(x, "signal");
```

```
use
outlet_new(x, "multichannelsignal");
```

## 19.4 Channel Counting in the Perform Method

Most of what you have to do is related to being a bit more careful about what you might previously have been able to assume about your perform method.

The prototype for your MSP perform method looks like this:

```
void myobject_perform64(t_myobject **x, t_object *dsp64, double **ins, long numins, double **outs, long numouts,
    long sampleframes, long flags, void *userparam);
```

Let's say your object will have one multi-channel signal input and one multi-channel signal output. As you can probably guess, the numins parameter to the perform method will be the count of channels in the input and the numouts parameter will be the count of channels in the output. Now let's consider some other cases because once we have more than one inlet and/or outlet, things get trickier.

Consider an object with two multi-channel inputs. You don't know in advance how many channels will be in each signal connected to your object. It could be 1 (if the user connects an old-style patch cord). It could be 100. It could be there is no connection at all to your object. What do you do?

If you're in this situation, you'll need to ask MSP how many channels are in each input in your `dsp64` method, which is called before the DSP is turned on. The prototype for your `dsp64` method looks like this:

```
void myobject_dsp64(t_myobject **x, t_object *dsp64, short *count, double samplerate, long maxvectorsize, long
    flags);
```

The dsp64 object passed to this method can be used to interrogate the number of channels in each of your object's inlets via the getnuminputchannels method. If your object has two inlets, here is how you can find out how many input channels each inlet has:

```
void myobject_dsp64(t_myobject *x, t_object *dsp64, short *count, double samplerate, long maxvectorsize, long flags)
{
    long leftinletchannelcount, rightinletchannelcount;
    leftinletchannelcount = (long)object_method(dsp64, gensym("getnuminputchannels"), x, 0);
    rightinletchannelcount = (long)object_method(dsp64, gensym("getnuminputchannels"), x, 1);
}
```

Note that an unconnected inlet has one channel, which, as has always been in the case in MSP, will be a signal containing all zeroes.

You might want to store these channel count values in your object so you can make use of them in your perform method. Then you'll know how to interpret the ins array of audio buffers you receive.

## 19.5 Specifying Output Channel Counts

The channel count for a multi-channel signal outlet is determined when MSP is building the DSP chain with the signal compiler. This means it can change each time the user turns the audio on if the graph has changed.

In MC, it's important to remember that outlets, not inlets, determine signal channel counts. You report the number of channels your object's outlets will have by supporting the multichanneloutputs method.

```
class_addmethod(c, (method)myobject_multichanneloutputs, "multichanneloutputs", A_CANT, 0);
...
long myobject_multichanneloutputs(t_myobject *x, long outletindex)
{
    return 4;
}
```

The above creates an outlet with four channels. Every time.

## 19.6 How to Auto-Adapt

If your object has defined multi-channel outlets it may receive the inputchanged message when the MSP signal compiler runs. This notifies your object how many channels are going to be sent to one of your object's inlets. You don't have to implement an inputchanged method, but you can use the information it provides to auto-adapt your object's number of output channels in one or more of your multi-channel signal outlets. The MC Wrapper performs auto-adapting when the user does not specify a fixed number of channels. For example, if "mc.cycle~ 440 @chans 64" is connected to the input of mc.\*~, the wrapper will create 64 instances of a \*~ object, one to multiply the output of each of the 64 cycle~ objects in the mc.cycle~.

Auto-adapting is a "conversational" protocol that involves both the inputchanged and multichanneloutputs methods.

Here's how it works:

First, implement an the inputchanged method in addition to a multichannel outputs method:

```
class_addmethod(c, (method)myobject_inputchanged, "inputchanged", A_CANT, 0);
...
long myobject_inputchanged(t_myobject *x, long index, long count)
{
    // do something here and return true or false
}
```

Index is the inlet index of your object (with the leftmost being zero) and count is the number of audio channels in that particular inlet.

Your object's `inputchanged` method should return true if its idea of how many outputs one of your outlets may be changing based on the information just received. It should return false if it is not going to change. Returning false is polite and optimizes the speed of compiling the signal chain.

You should also store the count you receive somewhere in your object if you are going to use it to modify the count of output channels. After you return true from the `inputchanged` method your object's `multichanneloutputs` method will be called for every `multichannelsignal` outlet your object has created. You can then return the new channel count based on the information received in the `inputchanged` method.

Here's an illustration of the auto-adapting protocol with a simple example of an object with one inlet and one `multichannelsignal` outlet. First, the object will receive the `inputchanged` method:

```
long myobject_inputchanged(t_myobject **x, long index, long count)
{
    if (count != x->m_inputcount) {
        x->m_inputcount = count;
        return true;
    }
    else
        return false;
}
long myobject_multichanneloutputs(t_myobject **x, long index)
{
    return x->m_inputcount;
}
```

Note that the `inputchanged` and `multichanneloutputs` methods will be sent to your object before the `dsp64` method.

It's a good practice to use the `getnuminputchannels` technique inside your object's `dsp64` method as demonstrated above even if you support the `inputchanged` method. The signal compiler is not guaranteed to send the `inputchanged` message to your object in all cases — for example, it may not send `inputchanged` if there is nothing connected to one of your inlets, so for determining the output count you should assume one channel until `inputchanged` tells you something else. (Currently the `inputchanged` message is sent to objects with unconnected inlets, but this adds some overhead, so we're investigating whether it's always necessary.)

## 19.7 How Many Output Channels Do I Have?

If you want to determine the count of signal output channels your object has for any of its multi- channel (or single-channel) outlets in your `dsp64` method, you can send the message `getnumoutputchannels` to the `dsp64` object:

```
long leftoutletchannelcount = (long) object_method(dsp64, gensym("getnumoutputchannels"), x, 0);
```

The `getnumoutputchannels` method of the `dsp64` object works the same as the `getnuminputchannels` method.

Here's something potentially unintuitive about the MC signal compiler: it always gives you the number of output channels you want, whether or not that's a good idea. Here's what this means. Suppose you have an object with two `multichannelsignal` outlets. In your `dsp64` method, you notice that one of these outlets is not connected to anything (its entry in the `count[]` array is zero). You might assume this means the output channel count is zero, or maybe one (as with the number of channels given to an unconnected inlet). However, in reality it will be the number of channels you returned for this outlet in your `multichanneloutputs` method. The principle that applies here, established since the first version of MSP, is that you should never have to modify the behavior of your `perform` method based on whether its outputs are connected. You could choose not to call `dsp_add64` in a case where none of your object's outlets were connected in which case your `perform` method won't be called. But if the `perform` method is going to be called, it will receive the number of outputs you request.

To summarize, if your object has two outlets and it has returned a value of 12 for each outlet in its `multichanneloutputs` method, the `perform` method will receive a total of 24 output channels (in other words, the `numouts` parameter will be 24).

## 19.8 Handling MC Signals in Traditional MSP Objects

When an MC signal is connected to a traditional MSP object, then only the first channel of a multi-channel patch cord is handed to the object.

If you want to modify this behavior and receive all of the channels the you must supply the Z\_MC\_INLETS flag. Having supplied this flag, a user can connect single-channel patch cords, multi-channel patch cords, or both – and you'll have to make the best out of the situation.

## 19.9 Additional Help File Support

In addition to the helpfile tab argument in objectmappings mentioned above, you may wish to provide additional help patchers for your object.

When a user asks to open a help patcher for an object and there is only one help patcher available then that one help patcher is opened immediately. If a user asks to open a help patcher for an object with multiple help patchers then a menu is provided and the user selects the appropriate help patcher. Support for multiple help patchers was added in Max 8 to support MC. Specifically, the features of the MC wrapper have their own help patcher which can then be shared across multiple objects.

As an example, the mc.limi~ object uses the wrapper. To make the shared wrapper help patcher available as an option for mc.limi~, the following line is placed in an init file:

```
max classnamehelpcategory mc.limi~ mcwrap;
```

The category mcwrap is defined with a line also in the init file which looks like this:

```
max categoryhelp mcwrap mcwrapper-group "MC Wrapper Features Help";
```

The first argument to "categoryhelp" is the name of the category to create. The second argument is the name of a help patcher in the searchpath (minus its suffix). The third argument is the string that will appear in the menu.

To see the existing categories and mappings, look at the file named "mc-helpconfig.txt" inside the application bundle's init folder.

## 19.10 Examples

The first example is a signal visualizer called gridmeter~. It is included to demonstrate just two changes for objects receiving multi-channel inputs:

- First, the object checks the value returned by the getnuminputchannels method in the dsp64 method. This permits it to know how many channels to paint in the grid.
- Second, it does not assume a specific count of channels in its perform method, which was typically the case with meter objects in MSP. Instead, it has a loop for each input channel up to the value of the numins parameter.

The second example is called mc.rotate~ and demonstrates how to implement the auto-adapting protocol. mc.rotate~ simply rotates all the channels in any multi-channel signal it receives by one, but it will produce the same number of output channels as the number of inputs it receives.

Finally, there is also the example of mc.pack~ which takes only the first channel of any connected multi-channel signal. But if you type mc.combine~ instead it uses all the channels. mc.pack~ and mc.combine~ are the same object, but behave differently based on the convention that something called "pack" doesn't produce more outputs than it has inlets. This demonstrates the use of the Z\_MC\_INLETS flag.



# Chapter 20

## Jitter Object Model

### 20.1 Jitter Object Model Basics

Jitter objects use an object model which is somewhat different than the one traditionally used for developing Max external objects. The first big difference between Jitter objects and traditional Max external objects is that Jitter objects don't have any notion of the patcher themselves. This allows for the flexible instantiation and use of Jitter objects from C, Java, JavaScript, as well as in the Max patcher. The use of these Jitter objects is exposed to the patcher with a Max "wrapper" object, which will be discussed in the following chapter.

In this chapter we'll restrict our discussion to the fundamentals of defining the Jitter object which can be used in any of these languages. While Jitter's primary focus is matrix processing and real-time graphics, these tasks are unrelated to the object model, and will be covered in later chapters on developing Matrix Operator (MOP) and OB3D objects. Like Max objects, Jitter objects are typically written in C. While C++ can be used to develop Jitter objects, none of the object oriented language features will be used to define your object as far as Jitter is concerned. Similar to C++ or Java objects, Jitter objects are defined by a class with methods and member variables - we will refer to the member variables as "attributes". Unlike C++ or Java, there are no language facilities that manage class definition, class inheritance, or making use of class instances. In Jitter this must all be managed with sets of standard C function calls that will define your class, exercise methods, and get and set object attributes.

Max and Jitter implement their object models by maintaining a registry of ordinary C functions and struct members that map to methods and attributes associated with names. When some other code wishes to make use of these methods or attributes, it asks the Jitter object to look up the method or attribute in its registry based on a name. This is called dynamic binding, and is similar to Smalltalk or Objective C's object model. C++ and Java typically make use of static binding — i.e. methods and member variables are resolved at compile time rather than being dynamically looked up at run time.

### 20.2 Defining a Jitter Class

A Jitter class is typically defined in a C function named something like `your_object_name_init()`. Class definition begins with a call to `jit_class_new()`, which creates a new class associated with a specified name, constructor, destructor, and size in bytes of the object as stored in a C structure. This is followed by calls to `jit_class_addmethod()` and `jit_class_addattr()`, which register methods and attributes with their corresponding names in the class. The class is finally registered with a call to `jit_class_register()`. A minimal example class definition is shown below:

```
typedef struct _jit_foo
```

```

{
    t_jit_object      ob;
    float            myval;
} t_jit_foo;
static t_jit_class *_jit_foo_class=NULL;
t_jit_err jit_foo_init(void)
{
    long attrflags=0;
    t_jit_object *attr;
    // create new class named "jit_foo" with constructor + destructor
    _jit_foo_class = jit_class_new("jit_foo", (method) jit_foo_new,
        (method) jit_foo_free, sizeof(t_jit_foo), 0L);
    // add method to class
    jit_class_addmethod(jit_foo_scream, "scream", A_DEFLONG, 0L);
    // define attribute
    attr = jit_object_new(          // instantiate an object
        _jit_sym_jit_attr_offset, // of class jit_attr_offset
        "myval",                // with name "myval"
        _jit_sym_float32,       // type float32
        attrflags,              // default flags
        (method)0L,              // default getter accessor
        (method)0L,              // default setter accessor
        calcoffset(_jit_foo, myval)); // byte offset to struct member
    // add attribute object to class
    jit_class_addattr(_jit_foo_class, attr);
    // register class
    jit_class_register(_jit_foo_class);
    return JIT_ERR_NONE;
}
// constructor
t_jit_foo *jit_foo_new(void)
{
    t_jit_foo *x;
    // allocate object
    if (x= jit_object_alloc(_jit_foo_class))
    {
        // if successful, perform any initialization
        x->myval = 0;
    }
    return x;
}
// destructor
void jit_foo_free(t_jit_foo *x)
{
    // would free any necessary resources here
}
// scream method
void jit_foo_scream(t_jit_foo *x, long i)
{
    post("MY VALUE IS %f! AND MY ARGUMENT IS %d", x->myval, i);
}

```

The above example has a constructor, `jit_foo_new()`; a destructor, `jit_foo_free()`; one 32 bit floating point attribute, `myval`, a member of the object struct accessed with default accessor methods; and a method `jit_foo_scream()`, which posts the current value of `myval` to the Max window.

## 20.3 Object Struct

Each instance of an object occupies some region of organized memory. The C structure that defines this organization of memory is typically referred to as the "object struct". It is important that the object struct always begin with an entry of type `t_jit_object`. It is within the `t_jit_object` where special information about the class is kept. The C structure can contain additional information, either exposed as attributes or not, but it is important that the size of the object struct does not exceed 16384 bytes. This means that it is not safe to define a large array as a struct entry if it will cause the size of the object struct to be larger than this limit. If additional memory is required, the object struct should contain a pointer to memory allocated from within the constructor, and freed within the destructor.

The class registration in the above code makes use of the object struct both to record in the class how large each object instance should be—i.e. `sizeof(t_jit_foo)` ; and at what byte offset in the object struct an attribute is located—i.e.

`calcoffset(t_jit_foo, myval)`. When methods of an object are called, the instance of the object struct is passed as the first argument to the C functions which define the object methods. This instance may be thought of as similar to the "this" keyword used in C++ and Java - actually the C++ and Java underlying implementation works quite similarly to what has been implemented here in pure C. Object struct entries may be thought of as similar to object member variables, but methods must be called via functions rather than simply dereferencing instances of the class as you might do in C++ or Java. The list of object methods and other class information is referenced by your object's `t_jit_object` entry.

## 20.4 Constructor/Destructor

The two most important methods that are required for all objects are the constructor and the destructor. These are typically named `your_object_name_new()`, and `your_object_name_free()`, respectively. It is the constructor's responsibility to allocate and initialize the object struct and any additional resources the object instance requires. The object struct is allocated via `jit_object_alloc()`, which also initializes the `t_jit_object` struct entry to point at your relevant class information. The class information resides in your global class variable, e.g. `_jit_foo_class`, which you pass as an argument to `jit_object_alloc()`. This allocation does not, however initialize the other struct entries, such as "myval", which you must explicitly initialize if your allocation is successful. Note that because the constructor allocates the object instance, no object instance is passed as the first argument to the function which defines the constructor, unlike other object methods.

The constructor also has the option of having a typed argument signature with the same types as defined in the Writing Max Externals documentation—i.e. `A_LONG`, `A_FLOAT`, `A_SYM`, `A_GIMME`, etc. Typically, Jitter object constructors either have no arguments or use the `A_GIMME` typed argument signature.

In earlier versions of Jitter, the constructors were often specified as private and "untyped" using the `A_CANT` type signature. While this obsolete style of an untyped constructor will work for the exposure of a Jitter class to the patcher and C, it is now discouraged, as there must be a valid type signature for exposure of a class to Javascript or Java, though that signature may be the empty list.

It is the destructor's responsibility to free any resources allocated, with the exception of the object struct itself. The object struct is freed for you after your destructor exits.

## 20.5 Methods

You can define additional methods using the `jit_class_adDMETHOD()` function. This example defines the scream method associated with the function `jit_foo_scream()`, with no additional arguments aside from the standard first argument of a pointer to the object struct. Just like methods for ordinary Max objects, these methods could have a typed argument signature with the same types as defined in the Writing Max Externals documentation — i.e. `A_LONG`, `A_FLOAT`, `A_SYM`, `A_GIMME`. Typically in Jitter objects, public methods are specified either without arguments, or use `A_GIMME`, or the low priority variants, `A_DEFER_LOW`, or `A_USURP_LOW`, which will be discussed in following chapters. Private methods, just like their Max equivalent should be defined as untyped, using the `A_CANT` type signature. Object methods can be called from C either by calling the C function directly, or by using `jit_object_method()` or `jit_object_method_typed()`.

For example, the following calls that relate to the above jit\_foo example are equivalent:

```
// call scream method directly
jit_foo_scream(x, 74);
// dynamically resolve and call scream method
jit_object_method(x, gensym("scream"), 74);
// dynamically resolve and call scream method with typed atom arguments
t_atom a[1];
jit_atom_setlong(a, 74);
jit_object_method_typed(x, gensym("scream"), 1, a, NULL);
```

What the `jit_object_method()` and `jit_object_method_typed()` functions do is look up the provided method symbol in the object's class information, and then calls the corresponding C function associated with the provided symbol. The difference between `jit_object_method()` and `jit_object_method_typed()` is that `jit_object_method()` will not require that the

method is typed and public, and blindly pass all of the arguments following the method symbol on to the corresponding method. For this reason, it is required that you know the signature of the method you are calling, and pass the correct arguments. This is not type checked at compile time, so you must be extremely attentive to the arguments you pass via `jit_object_method()`. It is also possible for you to define methods which have a typed return value with the `A_GIMMEBACK` type signature. When calling such methods, the final argument to `jit_object_method_typed()`, should point to a `t_atom` to be filled in by the callee. This and the subject of "typed wrappers" for exposing otherwise private methods to language bindings that require typed methods (e.g. Java/JavaScript) will be covered in a later chapter.

## 20.6 Attributes

You can add attributes to the class with `jit_class_addattr()`. Attributes themselves are Jitter objects which share a common interface for getting and setting values. While any class which conforms to the attribute interface could be used to define attributes of a given class, there are a few common classes which are currently used: `jit_attr_offset()`, which specifies a scalar attribute of a specific type (char, long, float32, float64, symbol, or atom) at some byte offset in the object struct; `jit_attr_offset_array()` which specifies an array (vector) attribute of a specific type (char, long, float32, float64, symbol, or atom) at some byte offset in the object struct; and `jit_attribute`, which is a more generic attribute object that can be instantiated on a per object basis. We will not document the usage of `jit_attribute` at this time. The constructor for the class `jit_attr_offset()` has the following prototype:

```
t_jit_object *jit_attr_offset_new(char *name, t_symbol *type, long flags,
method mget, method mset, long offset);
```

When this constructor is called via `jit_object_new()`, additionally the class name, `_jit_sym_jit_attr_offset` (a global variable equivalent to `gensym("jit_attr_offset")`) must be passed as the first parameter, followed by the above arguments, which are passed on to the constructor. The name argument specifies the attribute name as a null terminated C string. The type argument specifies the attribute type, which may be one of the following symbols: `_jit_sym_char`, `_jit_sym_long`, `_jit_sym_float32`, `_jit_sym_float64`, `_jit_sym_symbol`, `_jit_sym_atom`, `_jit_sym_object`, or `_jit_sym_pointer`. The latter two are only useful for private attributes as these types are not exposed to, or converted from Max message atom values.

The flags argument specifies the attribute flags, which may be a bitwise combination of the following constants:

```
#define JIT_ATTR_GET_OPAQUE          0x00000001 // cannot query
#define JIT_ATTR_SET_OPAQUE          0x00000002 // cannot set
#define JIT_ATTR_GET_OPAQUE_USER      0x00000010 // user cannot query
#define JIT_ATTR_SET_OPAQUE_USER      0x00000020 // user cannot set
#define JIT_ATTR_GET_DEFER           0x00010000 // (deprecated)
#define JIT_ATTR_GET_USURP            0x00020000 // (deprecated)
#define JIT_ATTR_GET_DEFER_LOW        0x00040000 // query in low priority
#define JIT_ATTR_GET_USURP_LOW        0x00080000 // query in low, usurping
#define JIT_ATTR_SET_DEFER            0x01000000 // (deprecated)
#define JIT_ATTR_SET_USURP            0x02000000 // (deprecated)
#define JIT_ATTR_SET_DEFER_LOW        0x04000000 // set at low priority
#define JIT_ATTR_SET_USURP_LOW        0x08000000 // set at low, usurping
```

Typically attributes in Jitter are defined with flags `JIT_ATTR_GET_DEFER_LOW`, and `JIT_ATTR_SET_USURP_LOW`. This means that multiple queries from the patcher will generate a response for each query, and that multiple attempts to set the value at high priority will collapse into a single call with the last received value. For more information on defer and usurp, see the chapter on Jitter scheduling issues.

The mget argument specifies the attribute "getter" accessor method, used to query the attribute value. If this argument is zero (NULL), then the default getter accessor will be used. If you need to define a custom accessor, it should have a prototype and form comparable to the following custom getter:

```
t_jit_err jit_foo_myval_get(t_jit_foo *x, void *attr, long *ac, t_atom **av)
{
    if ((*ac)&&(*av)) {
        //memory passed in, use it
    } else {
        //otherwise allocate memory
        *ac = 1;
        if (!(*av = jit_getbytes(sizeof(t_atom)*(*ac)))) {

```

```

        *ac = 0;
        return JIT_ERR_OUT_OF_MEM;
    }
}
jit_atom_setfloat(*av,x->myval);
return JIT_ERR_NONE;
}

```

Note that getters require memory to be allocated, if there is not memory passed into the getter. Also the attr argument is the class' attribute object and can be queried using jit\_object\_method() for things like the attribute flags, names, filters, etc.. The mset argument specifies the attribute "setter" accessor method, used to set the attribute value. If this argument is zero (NULL), then the default setter accessor will be used. If we need to define a custom accessor, it should have a prototype and form comparable to the following custom setter:

```

t_jit_err jit_foo_myval_set(t_jit_foo *x, void *attr, long ac, t_atom *av)
{
    if (ac&&av) {
        x->myval = jit_atom_getfloat(av);
    } else {
        // no args, set to zero
        x->myval = 0;
    }
    return JIT_ERR_NONE;
}

```

The offset argument specifies the attribute's byte offset in the object struct, used by default getters and setters to automatically query and set the attribute's value. If you have both custom accessors, this value is ignored. This can be a useful strategy to employ if you wish to have an object attribute that does not correspond to any actual entry in your object struct. For example, this is how we implement the time attribute of jit.movie — i.e. it uses a custom getter and setter which make QuickTime API calls to query and set the current movie time, rather than manipulating the object struct itself, where no information about movie time is actually stored. In such an instance, you should set this offset to zero.

After creating the attribute, it must be added to the Jitter class using the [jit\\_class\\_addattr\(\)](#) function:

```
t_jit_err jit_class_addattr(void *c, t_jit_object *attr);
```

To put it all together: to define a jit\_attribute\_offset() with the custom getter and setter functions defined above, you'd make the following call:

```
long attrflags = JIT_ATTR_GET_DEFER_LOW | JIT_ATTR_SET_USURP_LOW;
t_jit_object *attr = jit_object_new(_jit_sym_jit_attr_offset, "myval", _jit_sym_float32, attrflags,
    (method)jit_foo_myval_get, (method)jit_foo_myval_set, NULL);
jit_class_addattr(_jit_foo_class, attr);
```

And to define a completely standard jit\_attribute\_offset(), using the default getter and setter methods:

```
long attrflags = JIT_ATTR_GET_DEFER_LOW | JIT_ATTR_SET_USURP_LOW;
t_jit_object *attr = jit_object_new(_jit_sym_jit_attr_offset, "myval", _jit_sym_float32, attrflags,
    (method)NULL, (method)NULL, calcoffset(t_jit_foo, myval));
jit_class_addattr(_jit_foo_class, attr);
```

## 20.7 Array Attributes

Attributes can, in addition to referencing single values, also refer to arrays of data. The class `jit_attribute_offset_array` is used in this instance. The constructor for the class `jit_attr_offset_array()` has the following prototype:

```
t_jit_object *jit_attr_offset_array_new(char *name, t_symbol *type, long size,
    long flags, method mget, method mset, long offsetcount, long offset);
```

When this constructor is called via `jit_object_new()`, additionally the class name, `_jit_sym_jit_attr_offset_array()` (a global variable equivalent to `gensym("jit_attr_offset_array")`) must be passed as the first parameter, followed by the above arguments, which are passed on to the constructor.

The name, type, flags, mget, mset and offset arguments are identical to those specified above.

The size argument specifies the maximum length of the array (the allocated size of the array in the Jitter object struct). The offsetcount specifies the byte offset in the object struct, where the actual length of the array can be queried/set. This value should be specified as a long. This value is used by default getters and setters when querying and setting the attribute's value. As with the jit\_attr\_offset object, if you have both custom accessors, this value is ignored.

The following sample listing demonstrates the creation of a simple instance of the jit\_attr\_offset\_array() class for an object defined as:

```
typedef struct _jit_foo
{
    t_jit_object      ob;
    long              myarray[10]; // max of 10 entries in this array
    long              myarraycount; // actual number being used
} t_jit_foo;
long attrflags = JIT_ATTR_GET_DEFER_LOW | JIT_ATTR_SET_USURP_LOW;
t_jit_object *attr = jit_object_new(_jit_sym_jit_attr_offset_array, "myarray",
    _jit_sym_long, 10, attrflags, (method)0L, (method)0L,
    calcoffset(t_jit_foo, myarraycount), calcoffset(t_jit_foo, myarray));
jit_class_addattr(_jit_foo_class, attr);
```

## 20.8 Attribute Notification

Although the subject of object registration and notification will be covered in greater depth in a forthcoming chapter, it bears noting that attributes of all types (e.g. jit\_attr\_offset, jit\_attr\_offset\_array and jit\_attribute) will, if registered, automatically send notifications to all attached client objects, each time the attribute's value is set.

# Chapter 21

## Jitter Max Wrappers

### 21.1 Max Wrapper Classes

In order to expose the Jitter object to the Max patcher, a Max "wrapper" class must be defined. For simple classes, this is largely facilitated by a handful of utility functions that take a Jitter class and create the appropriate wrapper class with default functionality. However, there are occasions which warrant additional intervention to achieve special behavior, such as the use of additional inlets and outlets, integrating with MSP, converting matrix information to and from Max lists, etc. The first Max wrapper class we'll demonstrate won't have any extra complication beyond simply containing a basic Jitter class.

In general it is preferable to design the Jitter class so that it knows nothing about the Max patcher, and that any logic necessary to communicate with the patcher is maintained in the Max wrapper class. In situations where this might seem difficult, this can typically be accomplished by making special methods in the Jitter class that are only meant to be called by the Max wrapper, or by using Jitter's object notification mechanism, which we'll discuss in a future chapter. Below is the minimal Max wrapper class for the minimal Jitter class shown in the last chapter.

```
typedef struct _max_jit_foo
{
    t_object          ob;
    void             *obex;
} t_max_jit_foo;
void *class_max_jit_foo;
void ext_main(void *r)
{
    void *p,*q;
    // initialize the Jitter class
    jit_foo_init();
    // create the Max class as documented in Writing Max Externals
    setup(&class_max_jit_foo,
        (method) max_jit_foo_new,
        (method) max_jit_foo_free,
        (short) sizeof(t_max_jit_foo),
        0L, A_GIMME, 0);
    // specify a byte offset to keep additional information
    p = max_jit_classex_setup(calcoffset(t_max_jit_foo, obex));
    // look up the Jitter class in the class registry
    q = jit_class_findbyname gensym("jit_foo");
    // wrap the Jitter class with the standard methods for Jitter objects
    max_jit_classex_standard_wrap(p, q, 0);
    // add an inlet/outlet assistance method
    addmess((method)max_jit_foo_assist, "assist", A_CANT,0);
}
void max_jit_foo_assist(t_max_jit_foo **x, void *b, long m, long a, char *s)
{
    // no inlet/outlet assistance
}
void max_jit_foo_free(t_max_jit_foo **x)
{
```

```

// lookup the internal Jitter object instance and free
jit_object_free(max_jit_obex jitob_get(x));
// free resources associated with the obex entry
max_jit_obex_free(x);
}
void *max_jit_foo_new(t_symbol *s, long argc, t_atom *argv)
{
    t_max_jit_foo *x;
    long attrstart;
    void *o;
    // create the wrapper object instance based on the
    // max wrapper class, and the jitter class
    if (x = (t_max_jit_foo *)max_jit_obex_new(class_max_jit_foo,
        gensym("jit_foo")))
    {
        // add a general purpose outlet (rightmost)
        max_jit_obex_dumpout_set(x, outlet_new(x,0L));
        // get normal args if necessary
        attrstart = max_jit_attr_args_offset(argc,argv);
        // instantiate Jitter object
        if (o = jit_object_new(gensym("jit_foo")))
        {
            // set internal jitter object instance
            max_jit_obex jitob_set(x,o);
            // process attribute arguments
            max_jit_attr_args(x,argc,argv);
        }
        else
        {
            // couldn't instantiate, clean up and report an error
            freeobject((void *)x);
            x = NULL;
            error("jit.foo: out of memory");
        }
    }
    return (x);
}

```

## 21.2 Object Struct

The first thing you must do is define your Max class object struct. As is typical, for standard Max objects the first entry of the object struct must be of type `t_object`; for UI objects, it must be of type `t_jbox`; for MSP objects, it must be of type `t_pxobject`; and for MSP UI objects, it must be of type `t_pxjbox`. For more information on these different Max object types, please consult the Max developer documentation. Jitter objects can be wrapped within any of these object types.

You also need to define a pointer to point to extra information and resources needed to effectively wrap your Jitter class. This is typically referred to as the "obex" data, and it is where Jitter stores things like attribute information, the general purpose "dumpout", the internal Jitter object instance, Matrix Operator resources for inlets/outlets, and other auxiliary object information that is not required in a simple Max object. As of Max 4.5 there is also the facility for making use of such additional object information for ordinary Max objects. At the time of this writing, such information is provided in the Pattr developer documentation, as it is relevant to the definition of object attributes, which may be stored and operated upon by the patcher attribute suite of objects.

## 21.3 Defining Your Max Class

In your Max class registration, which takes place in your external's `ext_main()` function, you should begin by calling your Jitter class's registration function, typically named something like `your_object_name_init()`. Then you should proceed to define the Max class's constructor, destructor, object struct size, and typed arguments as is typically accomplished for Max objects via the setup function. In order for your wrapper class to be able to find the obex data, you need to specify a byte offset where this pointer is located within each object instance and allocate the resource in which this is stored in your Max class. This is accomplished with the `max_jit_classex_setup()` function. You should then

look up the Jitter class via `jit_class_findbyname()`, and wrap it via the `max_jit_classex_standard_wrap()` function. The `max_jit_classex_standard_wrap()` function will add all typed methods defined in the Jitter class, as well getter and setter methods for attributes that are not opaque (i.e. private), and all the methods that are common to Jitter objects like `getattributes`, `getstate`, `summary`, `importattrs`, `exportattrs`, etc.

Now that you have wrapped the Jitter class, you can add any additional methods that you wish, such as your inlet/outlet assistance method, or something specific to the Max object. Like Jitter objects, you can also add methods which have defer or usurp wrappers, and these should be added via the `max_addmethod_defer_low()` or `max_addmethod_usurp_low()` functions, rather than simply using the traditional `addmess()` function. C

## 21.4 Constructor

Inside the Max object constructor, there are a few things which are different than building an ordinary Max external. If your object is to respond to attribute arguments, the constructor must be defined to take variable number of typed atom arguments, accomplished with the `A_GIMME` signature. You allocate your Max object with the `max_jit_obex_new()` function, instead of the traditional `newobject` function. You need to pass your Jitter class name to the `max_jit_obex_new()` function, which also allocates and initializes your obex data. If successful, you should proceed to add your general purpose "dumpout" outlet, used for returning attribute queries and other methods that provide information like `*jit.movie*`'s `framedump` method's frame number or `read` method success code, with the `max_jit_object_dumpout_set()` function. If your object is a Matrix Operator that calls `max_jit_mop_setup_simple()` you will not need to explicitly call `max_jit_object_dumpout_set()`, as `max_jit_mop_setup_simple()` calls `max_jit_object_dumpout_set()` internally.

You then allocate your Jitter object with `jit_object_new()`, and store it in your obex data via `max_jit_obex jitob_set()`. Note that this Jitter object instance can always be found with the function `max_jit_obex jitob_get()`. If you wish, prior to allocating your Jitter object, you can look at your non-attribute arguments first — those arguments up to the location returned by `max_jit_attr_args_offset()` — and make use of them in your Jitter object constructor. It is typical to process attribute arguments after you've allocated both the Max and Jitter object instances, with `max_jit_attr_args()`, which is passed the Max object instance. If you wanted to use the attribute arguments somehow in your Jitter object constructor, you would need to parse the attribute arguments yourself. If you are not able to allocate your Jitter object (as is the case if you have run out of memory or if Jitter is present but not authorized), it is important that you clean up your Max wrapper object, and return NULL.

## 21.5 Destructor

In your Max object destructor, you additionally need to free your internal Jitter object with `jit_object_free()`, and free any additional obex data with `max_jit_obex_free()`. Matrix operators will typically require that `max_jit_mop_free()` is called, to free the resources allocated for matrix inputs and outputs. If your object has attached to a registered object for notification via `jit_object_attach()`, you should detach from that object in your destructor using `jit_object_detach()` to prevent invalid memory accesses as the registered object might attempt to notify the memory of a now freed object. Object registration and notification is discussed in further detail in following chapters.

## 21.6 Dumpout

The general purpose outlet, also known as "dumpout", is automatically used by the Max wrapper object when calling attribute getters and several of the standard methods like `summary`, or `getattributes`. It is also available for use in any other Max method you want, most easily accessed with the `max_jit_obex_dumpout()` function that operates similar to `outlet_anything()`, but uses the max object pointer rather than the outlet pointer as the first argument. The outlet pointer which has been set in your constructor can be queried with the `max_jit_obex_dumpout_get()` function, and used in the standard outlet calls. However, it is recommended for routing purposes that any output through the dumpout outlet is a message beginning with a symbol, rather than simply a bang, int, or float. Therefore, `outlet_anything()` makes the most sense to use.

## 21.7 Additional inlets/outlets

To add additional inlets and outlets to your Max external, a few things should be noted. First, if your object is a Matrix Operator, matrix inlets and outlets will be added either through either the high level [max\\_jit\\_mop\\_setup\\_simple\(\)](#), or lower level [max\\_jit\\_mop\\_inputs\(\)](#) or [max\\_jit\\_mop\\_outputs\(\)](#) calls. These Matrix Operator functions will be covered in the chapter on Matrix Operators. Secondly, if your object is an MSP object, all signal inlets and outlets must be leftmost, and all non-signal inlets and outlets must be to the right of any single inlets or outlets—i.e. they cannot be intermixed. Lastly, additional inlets should use proxies (covered in detail in the Max developer documentation) so that your object knows which inlet a message has been received. This is accomplished with the [max\\_jit\\_obex\\_proxy\\_new\(\)](#) function. The inlet number is zero based, and you do not need to create a proxy for the leftmost inlet. Inside any methods which need to know which inlet the triggering message has been received, you can use the [max\\_jit\\_obex\\_inletnumber\\_get\(\)](#) function.

## 21.8 Max Wrapper Attributes

Sometimes you will need additional attributes which are specific to the Max wrapper class, but are not part of the internal Jitter class. Attributes objects for the Max wrapper class are defined in the same way as those for the Jitter class, documented in the previous chapter. However, these attributes are not added to the Max class with the [jit\\_class\\_addattr\(\)](#) function, but instead with the [max\\_jit\\_classex\\_addattr\(\)](#) function, which takes the classex pointer returned from [max\\_jit\\_classex\\_setup\(\)](#). Attribute flags, and custom getter and setter methods should be defined exactly as they would for the Jitter class.

## Chapter 22

# Matrix Operator QuickStart

The purpose of this chapter is to give a quick and high level overview of how to develop a simple Matrix Operator (MOP), which can process the matrix type most commonly used for video streams—i.e.

4 plane char data. For this task, we will use the jit.scalebias SDK example. More details such as how to make a Matrix Operator which deals with multiple types, plane count, dimensionality, inputs, outputs, etc. will appear in the following chapter. This chapter assumes familiarity with Jitter's multi-dimensional matrix representation and Matrix Operators used from the Max patcher, as discussed in the Jitter Tutorial, and as well as the preceding chapters on the Jitter object model and Max wrapper classes.

### 22.1 Defining the MOP Jitter Class

In the Jitter class definition, we introduce a few new concepts for Matrix Operators. In addition to the standard method and attribute definitions discussed in the Jitter object model chapter, you will want to define things like how many inputs and outputs the operator has, and what type, plane count, and dimension restrictions the operator has. These are accomplished by creating an instance of the `jit_mop` class, setting some state for the `jit_mop` object and adding this object as an adornment to your Jitter class. The following code segment references the `jit.scalebias` SDK example.

```
// create a new instance of jit_mop with 1 input, and 1 output
mop = jit_object_new(_jit_sym_jit_mop,1,1);
// enforce a single type for all inputs and outputs
jit_mop_single_type(mop,_jit_sym_char);
// enforce a single plane count for all inputs and outputs
jit_mop_single_planeCount(mop,4);
// add the jit_mop object as an adornment to the class
jit_class_addAdornment(_jit_scalebias_class,mop);
```

You create your `jit_mop` instance in a similar fashion to creating your attribute instances, using `jit_object_new()`. The `jit_mop` constructor has two integer arguments for inputs and outputs, respectively. By default, each MOP input and output is unrestricted in plane count, type, and dimension, and also are linked to the plane count, type, and dimensions of the first (i.e. leftmost) input. This default behavior can be overridden, and this simple 4 plane, char type, `jit.scalebias` example enforces the corresponding type and plane count restrictions via the `jit_mop_single_type()` and `jit_mop_single_planeCount()` utility functions. For more information on the `jit_mop` class, please see the following chapter on MOP details and the Jitter API reference.

Once you have created your `jit_mop` instance, and configured it according to the needs of your object, you add it as an adornment to your Jitter class with the `jit_class_add_adornment()` function. Adornments are one way for Jitter objects to have additional information, and in some instances behavior, tacked onto an existing class. Adornments will be discussed in detail in a later chapter.

You also want to define your matrix calculation method, where most of the work of a Matrix Operator occurs, with the `jit_class_addmethod()` function as a private, untyped method bound to the symbol `matrix_calc`.

```
jit_class_addmethod(_jit_scalebias_class,
(method) jit_scalebias_matrix_calc,
"matrix_calc", A_CANT, 0L);
```

## 22.2 The Jitter Class Constructor/Destructor

You don't need to add anything special to your Matrix Operator's constructor or destructor, aside from the standard initialization and cleanup any Jitter object would need to do. Any internal matrices for input and outputs are maintained, and only required, by the Max wrapper's asynchronous interface. The Jitter MOP contains no matrices for inputs and outputs, but rather expects that the matrix calculation method is called with all inputs and outputs synchronously. When used from languages like C, Java, and JavaScript, it is up to the programmer to maintain and provide any matrices which are being passed into the matrix calculation method.

## 22.3 The Matrix Calculation Method

The most important method for Matrix Operators, and the one in which the most work typically occurs is in the matrix calculation, or "matrix\_calc" method, which should be defined as a private, untyped method with the `A_CANT` type signature, and bound to the symbol "matrix\_calc". In this method your object receives a list of input matrices and output matrices to use in its calculation. You need to lock access to these matrices, inquire about important attributes, and ensure that any requirements with respect to type, plane count, or dimensionality for the inputs are met before actually processing the data, unlocking access to the matrices and returning. It should be defined as in the following example.

```
t_jit_err jit_scalebias_matrix_calc(t_jit_scalebias **,
    void *inputs, void *outputs)
{
    t_jit_err err=JIT_ERR_NONE;
    long in_savelock,out_savelock;
    t_jit_matrix_info in_minfo,out_minfo;
    char *in_bp,*out_bp;
    long i,dimcount,planecount,dim[JIT_MATRIX_MAX_DIMCOUNT];
    void *in_matrix,*out_matrix;
    // get the zeroth index input and output from
    // the corresponding input and output lists
    in_matrix = jit_object_method(inputs,_jit_sym_getindex,0);
    out_matrix = jit_object_method(outputs,_jit_sym_getindex,0);
    // if the object and both input and output matrices
    // are valid, then process, else return an error
    if ((x&&in_matrix&&out_matrix)
    {
        // lock input and output matrices
        in_savelock =
            (long) jit_object_method(in_matrix,_jit_sym_lock,1);
        out_savelock =
            (long) jit_object_method(out_matrix,_jit_sym_lock,1);
        // fill out matrix info structs for input and output
        jit_object_method(in_matrix,_jit_sym_getinfo,&in_minfo);
        jit_object_method(out_matrix,_jit_sym_getinfo,&out_minfo);
        // get matrix data pointers
        jit_object_method(in_matrix,_jit_sym_getdata,&in_bp);
        jit_object_method(out_matrix,_jit_sym_getdata,&out_bp);
        // if data pointers are invalid, set error, and cleanup
        if (!in_bp) { err=JIT_ERR_INVALID_INPUT; goto out; }
        if (!out_bp) { err=JIT_ERR_INVALID_OUTPUT; goto out; }
        // enforce compatible types
        if ((in_minfo.type!=_jit_sym_char) ||
            (in_minfo.type!=out_minfo.type))
        {
            err=JIT_ERR_MISMATCH_TYPE;
            goto out;
        }
        // enforce compatible planecount
        if ((in_minfo.planecount!=4) ||
            (out_minfo.planecount!=4))
        {
            err=JIT_ERR_MISMATCH_PLANE;
            goto out;
        }
        // get dimensions/planecount
        dimcount = out_minfo.dimcount;
        planecount = out_minfo.planecount;
        for (i=0;i<dimcount;i++)
        {
            // if input and output are not matched in
```

```

    // size, use the intersection of the two
    dim[i] = MIN(in_minfo.dim[i],out_minfo.dim[i]);
}
// calculate, using the parallel utility function to
// call the calculate_ndim function in multiple
// threads if there are multiple processors available
jit_parallel_ndim_simplecalc(
    (method) jit_scalebias_calculate_ndim,
    x, dimcount, dim, planecount,
    &in_minfo, in_bp, &out_minfo, out_bp,
    0, 0);
} else {
    return JIT_ERR_INVALID_PTR;
}
out:
// restore matrix lock state to previous value
jit_object_method(out_matrix,_jit_sym_lock,out_savelock);
jit_object_method(in_matrix,_jit_sym_lock,in_savelock);
return err;
}

```

## 22.4 Processing N-Dimensional Matrices

Since Jitter supports the processing of N-dimensional matrices where N can be any number from 1 to 32, most Matrix Operators are designed with a recursive function that will process the data in some lower dimensional slice, most often 2 dimensional. The recursive function that does this is typically named myobject\_calculate\_ndim(), and is called by your matrix\_calc method either directly or via one of the parallel processing utility functions, which are discussed in a future chapter.

It is out of the scope of this documentation to provide a detailed tutorial on fixed point or pointer arithmetic, both of which are used in this example. The code increments a pointer through the matrix data, scaling each planar element of each matrix cell by some factor and adding some bias amount. This is done with fixed point arithmetic (assuming an 8bit fractional component), since a conversion from integer to floating point data and back is an expensive operation. The jit.scalebias object also has two modes, one which sums the planes together, and one which processes each plane independently. You can improve performance by case handling on a per row, rather than per cell basis, and reduce your code somewhat by case handling on a per row, rather than per matrix basis. While a slight performance increase could be made by handling on a per matrix basis, per row is usually a decent point at which to make such an optimization trade off.

```

// recursive function to handle higher dimension matrices,
// by processing 2D sections at a time
void jit_scalebias_calculate_ndim(t_jit_scalebias *x,
    long dimcount, long *dim, long planecount,
    t_jit_matrix_info *in_minfo, char *bip,
    t_jit_matrix_info *out_minfo, char *bop)
{
    long i,j,width,height;
    uchar *ip,*op;
    long ascale,rscale,gscale,bscale;
    long abias,rbias,gbias,bbias,sumbias;
    long tmp;
    if (dimcount<1) return; //safety
    switch(dimcount)
    {
        case 1:
            // if only 1D, interpret as 2D, falling through to 2D case
            dim[1]=1;
        case 2:
            // convert floating point scale factors to a fixed point int
            ascale = x->ascale*256.;
            rscale = x->rscale*256.;
            gscale = x->gscale*256.;
            bscale = x->bscale*256.;
            // convert floating point bias values to a fixed point int
            abias = x->abias*256.;
            rbias = x->rbias*256.;
            gbias = x->gbias*256.;
            bbias = x->bbias*256.;
            // for efficiency in sum mode (1), make a single bias value

```

```

sumbias = (x->abias+x->rbias+x->gbias+x->bbias)*256.;
width  = dim[0];
height = dim[1];
// for each row
for (i=0;i<height;i++)
{
    // increment data pointers according to byte strid
    ip = bip + i*in_minfo->dimstride[1];
    op = bop + i*out_minfo->dimstride[1];
    switch (x->mode) {
        case 1:
            // sum together, clamping to the range 0-255
            // and set all output planes
            for (j=0;j<width;j++) {
                tmp = (long)(*ip++)*ascale;
                tmp += (long)(*ip++)*rscale;
                tmp += (long)(*ip++)*gscale;
                tmp += (long)(*ip++)*bscale;
                tmp = (tmp>>8L) + sumbias;
                tmp = (tmp>255)?255:((tmp<0)?0:tmp);
                *op++ = tmp;
                *op++ = tmp;
                *op++ = tmp;
                *op++ = tmp;
            }
            break;
        default:
            // apply to each plane individually
            // clamping to the range 0-255
            for (j=0;j<width;j++) {
                tmp = (((long)(*ip++)*ascale)>>8L)+abias;
                *op++ = (tmp>255)?255:((tmp<0)?0:tmp);
                tmp = (((long)(*ip++)*rscale)>>8L)+rbias;
                *op++ = (tmp>255)?255:((tmp<0)?0:tmp);
                tmp = (((long)(*ip++)*gscale)>>8L)+gbias;
                *op++ = (tmp>255)?255:((tmp<0)?0:tmp);
                tmp = (((long)(*ip++)*bscale)>>8L)+bbias;
                *op++ = (tmp>255)?255:((tmp<0)?0:tmp);
            }
            break;
        }
    }
break;
default:
    // if processing higher dimension than 2D,
    // for each lower dimensioned slice, set
    // base pointer and recursively call this function
    // with decremented dimcount and new base pointers
    for (i=0;i<dim[dimcount-1];i++)
    {
        ip = bip + i*in_minfo->dimstride[dimcount-1];
        op = bop + i*out_minfo->dimstride[dimcount-1];
        jit_scalebias_calculate_ndim(x,dimcount1,
                                      dim,planecount,in_minfo,ip,out_minfo,op);
    }
}
}

```

Rather than using multidimensional arrays, Jitter matrix data is packed in a single dimensional array, with defined byte strides for each dimension for greatest flexibility. This permits matrices to reference subregions of larger matrices, as well as support data that is not tightly packed. Therefore, rather than using multidimensional array syntax, this code uses pointer arithmetic to access each plane of each cell of the matrix, adding the corresponding byte strides to the base pointer for each dimension across which it is iterating. These byte strides are stored in the dimstride entry of the [t\\_jit\\_matrix\\_info](#) struct. Note that Jitter requires that planes within a cell, and cells across the first dimension (dim[0]) are tightly packed. The above code assumes that this is the case, using a simple pointer increment for each plane and cell, rather than looking up byte strides for dim[0].

## 22.5 Defining the MOP Max Wrapper Class

In order to use the MOP class in a Max patcher you need to make a Max wrapper class. In addition to the standard methods used to wrap any Jitter class, MOPs need to add special methods and information to the Max class. One of

the things that needs to happen is that the Max wrapper class needs to allocate and maintain instances of jit.matrix for each matrix input and output other than the leftmost input, to accommodate Max's asynchronous event model. In order to perform this maintenance, the Max wrapper class must have special methods and attributes for setting the type, plane count, dimensions, adaptability, and named references for the internal matrices. All of these messages are exclusive to the Max wrapper implementation, and are not used by the C, Java, or JavaScript usage of Matrix Operators. There are also common methods and attributes for the matrix output mode, and the jit\_matrix and bang messages, all of which are specific to the MOP's Max wrapper. These special attributes and methods are added by the `max_jit_classex_mop_wrap()` function, which should be called inside your Max external's `ext_main()` function, after calling `max_jit_classex_setup()` and `jit_class_findbyname()`, and before calling `max_jit_classex_standard_wrap()`. Several default methods and attributes can be overridden using the various flags that can be combined for the flags argument to `max_jit_classex_mop_wrap()`. These flags, which for most simple MOPs won't be necessary, are listed below.

```
#define MAX_JIT_MOP_FLAGS_OWN_ALL          0xFFFFFFFF
#define MAX_JIT_MOP_FLAGS_OWN_JIT_MATRIX    0x00000001
#define MAX_JIT_MOP_FLAGS_OWN_BANG          0x00000002
#define MAX_JIT_MOP_FLAGS_OWN_OUTPUTMATRIX  0x00000004
#define MAX_JIT_MOP_FLAGS_OWN_NAME          0x00000008
#define MAX_JIT_MOP_FLAGS_OWN_TYPE          0x00000010
#define MAX_JIT_MOP_FLAGS_OWN_DIM           0x00000020
#define MAX_JIT_MOP_FLAGS_OWN_PLANECOUNT   0x00000040
#define MAX_JIT_MOP_FLAGS_OWN_CLEAR         0x00000080
#define MAX_JIT_MOP_FLAGS_OWN_NOTIFY        0x00000100
#define MAX_JIT_MOP_FLAGS_OWN_ADAPT         0x00000200
#define MAX_JIT_MOP_FLAGS_OWN_OUTPUTMODE   0x00000400
```

## 22.6 The Max Class Constructor/Destructor

Inside your Max class' constructor you need to allocate the matrices necessary for the MOP inputs and outputs, the corresponding matrix inlets and outlets, process matrix arguments and other MOP setup. The `max_jit_mop_setup_simple()` function takes care of these functions and some of the other necessary tasks of wrapping your Jitter instance. As such, the use of this function simplifies your Jitter class wrapping even further for the simple case where no special behavior, incompatible with `max_jit_mop_setup_simple()` is required. Here is the constructor for the Max class of the jit.scalebias object.

```
void *max_jit_scalebias_new(t_symbol *s, long argc, t_atom *argv)
{
    t_max_jit_scalebias *x;
    void *o;
    if (x = (t_max_jit_scalebias *)
        max_jit_obex_new(
        max_jit_scalebias_class,
        gensym("jit_scalebias")))
    {
        // instantiate Jitter object
        if (o= jit_object_new(gensym("jit_scalebias")))
        {
            // handle standard MOP max wrapper setup tasks
            max_jit_mop_setup_simple(x,o,argc,argv);
            // process attribute arguments
            max_jit_attr_args(x,argc,argv);
        }
        else
        {
            error("jit.scalebias: could not allocate object");
            freeobject(x);
        }
    }
    return (x);
}
```

Below is the listing of the `max_jit_mop_setup_simple()` function, demonstrating the smaller pieces, it manages for you. If your object has special requirements, you can use whatever subset of the following function as necessary.

```
t_jit_err max_jit_mop_setup_simple(void **x, void **o, long argc, t_atom *argv)
{
    max_jit_obex_jitob_set(x,o);
    max_jit_obex_dumpout_set(x,outlet_new(x,NULL));
```

```
max_jit_mop_setup(x);
max_jit_mop_inputs(x);
max_jit_mop_outputs(x);
max_jit_mop_matrix_args(x,argc,argv);
return JIT_ERR_NONE;
}
```

In your Max class' destructor, you need to free the resources allocated for your MOP. This is accomplished with the `max_jit_mop_free()` function, which should be called before you free your internal Jitter instance, and your Max class' obex data. As an example, the `jit.scalebias` destructor is listed below.

```
void max_jit_scalebias_free(t_max_jit_scalebias *x)
{
    // free MOP max wrapper resources
    max_jit_mop_free(x);
    // lookup internal Jitter object instance and free
    jit_object_free(max_jit_obex_jitob_get(x));
    // free resources associated with obex entry
    max_jit_obex_free(x);
}
```

## Chapter 23

# Matrix Operator Details

The purpose of this chapter is to fill in the details of what a Matrix Operator is and how it works.

Matrix data in Jitter is typically considered raw data without respect to what the data represents. This permits simple fundamental operations to be applied to different sorts of data without needing to know any special information. For this reason most MOPs are general purpose. The jit.scalebias example from the preceding chapter could be considered video specific in its terminology, and type and plane count restrictions, but fundamentally it is just calculating a product and sum on each plane of an incoming matrix. In this chapter, we'll cover the details of how to configure MOP inputs and outputs, any attribute restrictions or linking for those inputs and outputs, what you must do in your matrix\_calc method and how you expose your MOP to the Max environment, overriding default behavior if necessary.

### 23.1 Defining the MOP Jitter Class

As discussed in the Matrix Operator Quick Start, for MOPs you must create an instance of jit\_mop with the jit\_object\_new() function and add it to your Jitter class as an adornment with the jit\_class\_addadornment() function. The jit\_mop object holds information such as how many inputs and outputs the object has, what types, plane count, and dimension counts are supported, and how inputs should respond to incoming matrices. This information is only relevant to wrappers of the Jitter object which actually maintain additional matrices for inputs and outputs, as is the case with the MOP Max wrapper class. When used from C, Java, or JavaScript, it is the the programmer's responsibility to pass in matrices that conform to any restrictions imposed by the MOP. An example of instantiating and adding the jit\_mop object is below.

```
// create a new instance of jit_mop with 1 input, and 1 output
mop = jit_object_new(_jit_sym_jit_mop,1,1);
// add jit_mop object as an adornment to the class
jit_class_addadornment(_jit_your_class,mop);
```

### 23.2 The jit\_mop\_io Object

Each instance of jit\_mop contains some number of inputs and outputs, specified by the input and output arguments to the constructor. For each of these inputs and outputs there is an instance of jit\_mop\_io which records information specific to that input or output, such as type, plane count, and dimension restrictions. You can access the input or output objects by calling the getinput or getoutput methods with an integer index argument as below:

```
input = jit_object_method(mop,_jit_sym_getinput,1);
output = jit_object_method(mop,_jit_sym_getoutput,1);
```

Once you have obtained references to these inputs or outputs, you may query or set the jit\_mop\_io attributes. The attributes typically configured are: types, which is a list of symbols of permitted types, the first of which being the default; mindim and maxdim, which are the minimum and maximum permitted sizes for each dimension; mindimcount and maxdimcount, which are the minimum and maximum permitted number of dimensions permitted; minplaneccount and maxplaneccount, which are the minimum and maximum number of planes permitted; typelink, which is the flag that determines if the I/O should change its type to whatever the leftmost incoming matrix is; dimlink, which is the flag that determines if the I/O should change its dimensions to whatever the leftmost incoming matrix is; and planelink, which is the flag that determines if the I/O should change its plane count to whatever the leftmost incoming matrix is.

### 23.3 Restricting Input/Output Attributes

By default, all types, dimensions and plane count are permitted, and all linking is enabled. If you wish your MOP to have some specific restrictions, or difference in linking behaviors for any input or output in particular, you can set the corresponding attributes. For example, to set the plane count to always be four planes, you would set both the minplaneccount and maxplaneccount attributes to 4, as below:

```
output = jit_object_method(mop, _jit_sym_getoutput, 1);
jit_attr_setlong(output, _jit_sym_minplaneccount, 4);
jit_attr_setlong(output, _jit_sym_maxplaneccount, 4);
```

The jit.scalebias example could have set the plane count using the minplaneccount and maxplaneccount attributes rather than calling the utility function [jit\\_mop\\_single\\_planeccount\(\)](#), which internally sets these attributes. A similar thing could be done to restrict type and dimensions. As for linking, if you wish to develop an object where the right hand input does not adapt to the size of the leftmost input, as is the case with jit.convolve, you would turn off the dimlink attribute, as below:

```
input2 = jit_object_method(mop, _jit_sym_getinput, 2);
jit_attr_setlong(input2, _jit_sym_dimlink, 0);
```

Similar could be done to remove type and plane count linking, and the utility functions [jit\\_mop\\_input\\_nolink\(\)](#) and [jit\\_mop\\_output\\_nolink\(\)](#) set all of these link attributes to false (zero).

### 23.4 The ioproc Function

For right hand matrix inputs, incoming data is typically copied by the MOP Max wrapper class. When an incoming matrix is received by the MOP Max wrapper class, a function called the ioproc is called, and the default ioproc copies the data, using the current input attributes (which might be linked to the lefthand input). The default ioproc can be overridden by calling the ioproc method followed by a function with the signature as listed below in the [jit\\_mop\\_ioproc\\_copy\\_adapt\(\)](#) function. The [jit\\_mop\\_ioproc\\_copy\\_adapt\(\)](#) function will always adapt to that inlet's incoming matrix attributes, as long as they don't conflict with any restrictions. The SDK project for jit.concat demonstrates the use of the [jit\\_mop\\_ioproc\\_copy\\_adapt\(\)](#) function.

```
t_jit_err jit_mop_ioproc_copy_adapt(void *mop, void *mop_io, void *matrix)
{
    void *m; // destination matrix
    t_jit_matrix_info info;
    // look up destination matrix from mop_io
    if (matrix&&(m= jit_object_method(mop_io, _jit_sym_getmatrix)))
    {
        // retrieve incoming matrix info
        jit_object_method(matrix, _jit_sym_getinfo, &info);
        //restrict matrix info based on mop_io attribtues
        jit_object_method(mop_io, _jit_sym_restrict_type, &info);
        jit_object_method(mop_io, _jit_sym_restrict_dim, &info);
        jit_object_method(mop_io, _jit_sym_restrict_planeccount, &info);
        // set destination matrix info
        jit_object_method(m, _jit_sym_setinfo, &info);
        // copy the data with the frommatrix method
        jit_object_method(m, _jit_sym_frommatrix, matrix, NULL);
    }
    return JIT_ERR_NONE;
}
```

## 23.5 Variable Inputs/Outputs

You can specify variable input/output MOPs with a negative argument for input and/or outputs when constructing your jit\_mop object. When using variable inputs and/or outputs, there is not a jit\_mop\_io for each input and/or output within your class definition, and therefore the template type, dim, planecount, and linking attributes are not settable. If anything but the default behavior is required, you must accomplish it in another way — for example, either by overriding the jit\_matrix method of the MOP Max wrapper class, or defining an mproc method to be called from within the standard jit\_matrix method of the MOP Max wrapper class. The jit.pack, jit.unpack, jit.scissors, and jit.glue objects are a few SDK examples of MOPs with variable inputs and outputs. More information on overriding the jit\_matrix, mproc, and other default methods of the MOP Max wrapper class is covered later in this chapter.

## 23.6 Adding jit\_mop as a Class Adornment

Once you have configured all of the inputs and outputs of your jit\_mop object, you must add your jit\_mop object to your Jitter class with the `jit_class_adornment()` function. Adornments can be queried from the Jitter class at any time by calling `jit_class_adornment_get()` with the Jitter class pointer and the class name of the adornment object, as demonstrated below.

```
// add jit_mop object as an adornment to the class
jit_class_adornment(_jit_your_class,mop);
// look up jit_mop adornment
mop = jit_class_adornment_get(_jit_your_class,_jit_sym_jit_mop);
```

## 23.7 The Matrix Calculation Method

The entry point of the MOP Jitter class is the matrix\_calc method, which is passed a list of matrices for the input, and a list of matrices for the output. It is not the responsibility of the matrix\_calc method to perform any copying and adaptation behavior, but rather simply ensure that the matrices are valid, compatible, and if so, process. Certain objects may modify the dim, type, or planecount of the output matrices — e.g. the SDK project, jit.thin. However, it is the calling party's responsibility to perform any copying and conformance to MOP I/O restrictions as defined by the jit\_mop\_io objects—i.e. either the Max wrapper class, or the C, Java, or Javascript code which calls the matrix\_calc method.

## 23.8 Accessing the Input and Output Lists

The input and output lists passed as arguments to your matrix\_calc method are Jitter objects, and pointers to the individual inputs and outputs are acquired by calling the getindex method with an integer argument specifying the zero based list index. The return values should be tested to make sure they are not null. For example:

```
// get the zeroth index input and output from
// the corresponding input and output lists
in_matrix  = jit_object_method(inputs,_jit_sym_getindex,0);
out_matrix = jit_object_method(outputs,_jit_sym_getindex,0);
// if the object and both input and output matrices
// are valid, then process, else return an error
if (x&&in_matrix&&out_matrix)
{
    // ... process data ...
} else {
    return JIT_ERR_INVALID_PTR;
}
```

Technically, you can also pass in an instance of jit\_matrix in place of a list for the input or output arguments, since jit\_matrix has a getindex method which returns the jit\_matrix instance. This is an example of dynamic binding at work.

Another example of dynamic binding inside the matrix\_calc method is that the list elements might be instances of jit\_mop\_io, rather than instances of jit\_matrix. However, since Jitter uses dynamic binding and the jit\_mop\_io object is a "decorator" class for jit\_matrix, all corresponding methods are passed on to the jit\_matrix referenced by the jit\_mop\_io. In fact, any Jitter objects which respond to the standard interface for jit\_matrix could be passed as inputs or outputs. If this seems confusing, you need not think about the underlying implementation further, but instead can assume that what is being passed in is simply an instance of jit\_matrix. After all it should behave like one, even if it is not.

## 23.9 Locking and Unlocking Matrices

Prior to working with a matrix, it is necessary to "lock" it so that the data and attributes will not be changed across the duration of the operation. This is accomplished by calling the jit\_matrix instance's lock method with an integer argument of 1 (true) to lock the matrix. You should store the current lock state to restore when you're done processing. The lock operation should be the first thing to do after ensuring that the matrix objects are not NULL. For example

```
// lock input and output matrices
in_savelock = (long) jit_object_method(in_matrix, _jit_sym_lock, 1);
out_savelock = (long) jit_object_method(out_matrix, _jit_sym_lock, 1);
// ... process data ...
out:
// restore matrix lock state to previous value
jit_object_method(out_matrix, _jit_sym_lock, out_savelock);
jit_object_method(in_matrix, _jit_sym_lock, in_savelock);
```

## 23.10 Retrieving Matrix Information

Once you have locked the matrices, you are ready to find out some information about them. This is accomplished by calling the getinfo method with a pointer to an instance of the `t_jit_matrix_info` struct. The `t_jit_matrix_info` struct contains several common attributes of the matrix and data organization of the matrix data, and is a useful way to obtain this information in one call, rather than querying each attribute individually. This information is typically tested to verify compatibility with any assumptions the matrix\_calc method needs to make (since this method might be called from C, Java, or Javascript, you cannot assume that the MOP Max wrapper will have enforced these assumptions). It is also used to perform the appropriate pointer arithmetic based on type, plane count, dimensions, and the byte stride of those dimensions, since higher dimensions may not be tightly packed. The `t_jit_matrix_info` struct is listed below:

```
typedef struct _jit_matrix_info
{
    long      size;          // in bytes (0xFFFFFFFF=UNKNOWN)
    t_symbol *type;          // primitive type
    long      flags;          // matrix flags: my data?, handle?
    long      dimcount;        // # of dimensions
    long      dim[JIT_MATRIX_MAX_DIMCOUNT]; // dimension sizes
    long      dimstride[JIT_MATRIX_MAX_DIMCOUNT]; // in bytes
    long      planecount;      // # of planes
} t_jit_matrix_info;
```

And here is an example of calling the getinfo method to fill out the `t_jit_matrix_info` struct:

```
// fill out matrix info structs for input and output
jit_object_method(in_matrix, _jit_sym_getinfo, &in_minfo);
jit_object_method(out_matrix, _jit_sym_getinfo, &out_minfo);
```

## 23.11 Retrieving the Data Pointer

The `t_jit_matrix_info` struct is the meta data, but the actual matrix data can be accessed by acquiring the data pointer. You accomplish this by calling the matrix's getdata method, passing in a pointer to a pointer. This pointer can be any type, but it is typically a char (or byte) pointer since you may need to perform bytewise pointer arithmetic depending on

the type and dimstride of your matrix. It is essential to verify that this pointer is valid before attempting to operate on the data, as demonstrated below.

```
// get matrix data pointers
jit_object_method(in_matrix,_jit_sym_getdata,&in_bp);
jit_object_method(out_matrix,_jit_sym_getdata,&out_bp);
// if data pointers are invalid, set error, and cleanup
if (!in_bp) { err=JIT_ERR_INVALID_INPUT; goto out;}
if (!out_bp) { err=JIT_ERR_INVALID_OUTPUT; goto out;}
```

## 23.12 Processing the Data

While it is possible to incorporate the data processing code inside the matrix\_calc method, it is typical to rely on other routines to accomplish the N dimensional processing through recursion, potentially dispatching to multiple processors. The N-dimensional recursive processing function (typically named myobject\_calculate\_ndim) is discussed in the next section. You should pass in to the calculate\_ndim function your object pointer, the overall dimension count, dimension sizes, planeCount to consider in your calculation, together with the necessary matrix info structs and data pointers for each input and output. You can call this method directly as is the case in the following code:

```
// call calculate_ndim function directly in current thread
jit_scalebias_calculate_ndim(x, dimcount, dim, planeCount,
    &in_minfo, in_bp, &out_minfo, out_bp);
```

Or you can call this method with the parallel processing utility functions provided with Jitter 1.5 to automatically dispatch the processing of large matrices across multiple processors when available. This figure illustrates the dispatching and calculating of the parallel processing utility:

The parallel processing is accomplished by breaking up the matrix into smaller matrices that each reference subregions of the original inputs and outputs. No new objects are created, but rather just additional `t_jit_matrix_info` structs and offset data pointers. Jitter 1.5 maintains a pool of worker threads for this purpose, so there is no thread creation overhead, but rather only some small thread synchronization overhead. Jitter 1.5 only dispatches across multiple threads when the data count is large enough to justify this thread synchronization overhead.

An important thing worth noting is that if your object performs some kind of spatial operation (e.g. convolution, rotation, scaling, etc.), you will either need to account for the matrix segmentation used by the parallel utilities or avoid using parallel processing and call directly in the current thread. Since the jit.scalebias example only processes one pixel at a time (i.e. a pointwise operation), it is inherently parallelizable, so it takes advantage of multiple processors as below:

```
// calculate, using the parallel utility function to
// call the calculate_ndim function in multiple
// threads if there are multiple processors available
jit_parallel_ndim_simplecalc2(
    (method) jit_scalebias_calculate_ndim,
    x, dimcount, dim, planeCount,
    &in_minfo, in_bp, &out_minfo, out_bp,
    0, 0 );
```

**Important Note:** If you aren't sure if your object is a pointwise operator, or don't fully understand how to make your algorithm parallelizable, you shouldn't use the parallel utility functions in your object. You should simply call the function directly.

## 23.13 Processing N-Dimensional Matrices

In the Matrix Operator Quick Start chapter, we discussed how to define a recursive function to process N-dimensional data in 2D slices, using the jit.scalebias object as an example. This example was restricted to processing four plane char data, but many Jitter objects work with any type of data and any plane count. In order to support all types and plane counts, there needs to be some case handling to know how to step through the data, and what type data to interpret as so that you can perform the appropriate operations. There are a number of ways to approach this logic, and decisions to

make with respect to optimization. All this case handling can be a bit cumbersome, so when initially developing objects, it probably makes sense for you to focus on a single type and plane count, and only after you've adequately defined your operation, attempt to make your code robust to process any type of data and consider optimization of certain cases. The use of C macros, or C++ templates might be useful things to explore for better code re-use. As for code optimization, typically a decent atomic element to try and optimize is the "innermost" loop, avoiding branch conditions where possible.

This function is at the heart of the logic you will add in your own custom object. Since there is no "right way" to process this data, we won't cover any more code listings for the recursive N-dimensional processing function. However, the SDK projects that are good examples include: jit.clip, which performs a planar independent, pointwise operation (limiting numbers to some specified range); jit.rgb2luma, which performs a planar dependent, pointwise operation (converting RGB color to luminance); and jit.transpose, which performs a planar independent, spatial operation (rows become columns). For more ideas about N-dimensional matrix processing, we would recommend reading one of the several books available on 2D signal processing and/or image processing. Most of these concepts are easily generalized to higher dimensions.

## 23.14 Defining the MOP Max Wrapper Class

MOP Max wrapper classes typically have a large amount of default behavior, as setup through the `max_jit_classex_mop_wrap` function, based on the `jit_mop` Jitter class adornment, and user specified flags. You can either override all of the default behavior or just specific features. If you wish to override all of the default behavior, you can use the flag `MAX_JIT_MOP_FLAGS_OWN_ALL`, when calling the `max_jit_classex_mop_wrap()` function. If you need to make use of the `jit_mop` adornment(), the `jit_mop` can be looked up by calling the `jit_class_adornment_get()` method on the Jitter class. The `jit_mop_io` inputs and outputs can be queried and their attributes inspected, similar to how they were set in the MOP Jitter class definition, described earlier in this chapter. Here is an example of how to look up the `jit_mop` adornment of the `jit.scalebias` object:

```
// look up jitter class by name
 jclass = jit_class_findbyname(gensym("jit_scalebias"));
// look up jit_mop adornment
mop = jit_class_adornment_get(jclass, _jit_sym_jit_mop);
```

## 23.15 Overriding the `jit_matrix` Method

By default, a `jit_matrix` method is added which automatically manages matrix copying and calculation based on the incoming data. Most typical MOPs simply use the default `jit_matrix` method. However there are instances where it is necessary to override the default MOP method to get special behavior, such as recording which matrix input data is being input to as is the case for the `jit.op` SDK example, or to do something other than standard copying and adaptation as is the case for the `jit.pack` or `jit.str.op` SDK examples, or to prevent any `jit_matrix` method at all, as is the case for the `jit.noise` SDK example. To prevent the default `jit_matrix` method from being defined, you can use the flag `MAX_JIT_MOP_FLAGS_OWN_JIT_MATRIX`, when calling the `max_jit_classex_mop_wrap()` function. To define your own `jit_matrix` method, you can add an `A_GIMME` method bound to the symbol `jit_matrix`, in your `ext_main()` function. Here's an example from `jit.op`:

```
// add custom jit_matrix method in ext_main()
address((method)max_jit_op_jit_matrix, "jit_matrix", A_GIMME, 0);
void max_jit_op_jit_matrix(t_max_jit_op *x, t_symbol *s, short argc,
                           t_atom *argv)
{
    if (max_jit_obex_inletnumber_get(x))
    {
        // if matrix is received in right input,
        // record to override float or int input
        x->last = OP_LAST_MATRIX;
    }
    // now pass on to the default jit_matrix method
    max_jit_mop_jit_matrix(x,s,argc,argv);
}
```

The `jit.pack` and `jit.str.op` examples are a bit more involved and also better illustrate the kinds of tasks the default `jit_matrix` method performs.

## 23.16 Overriding the bang and outputmatrix Methods

A MOP Max wrapper class typically has a bang and outputmatrix method. These two methods are typically equivalent, and by default, both send out the most recently calculated matrix output. Certain objects that don't have a matrix output, like the jit.3m SDK example, typically override these messages with their own bang and sometimes outputmatrix method. These methods can be overridden by using the `MAX_JIT_MOP_FLAGS_OWN_BANG` and `MAX_JIT_MOP_FLAGS_OWN_OUTPUTMATRIX` flags when calling the `max_jit_classex_mop_wrap()` function. These flags are typically both passed in together.

## 23.17 Overriding the name, type, dim, and planecount Attributes

For each input and output, other than the leftmost input, there is, by default, an attribute added to query and set that input or output's matrix attributes, including name, type, dim, and planecount. While overriding the default attribute behavior is conceivably necessary to perform very specialized behavior, it is not used by any of the SDK examples. To prevent the addition of the default attributes for name, type, dim, and planecount, you can use the `MAX_JIT_MOP_FLAGS_OWN_NAME`, `MAX_JIT_MOP_FLAGS_OWN_TYPE`, `MAX_JIT_MOP_FLAGS_OWN_DIM`, and `MAX_JIT_MOP_FLAGS_OWN_PLANECOUNT` flags when calling the `max_jit_classex_mop_wrap()` function. To define your own attributes, you would follow the same means of defining any attributes for a Max wrapper class with the appropriate attribute name you wish to override.

## 23.18 Overriding the clear and notify Methods

By default, a clear and a notify method are added. The default clear method clears each of the input and output matrices. The default notify method, `max_jit_mop_notify()`, is called whenever any of the matrices maintained by the MOP are changed. If it is necessary to respond to additional notifications, it is important to call the `max_jit_mop_notify` function so that the MOP can perform any necessary maintenance with respect to input and output matrices, as demonstrated by the jit.notify SDK example. These methods can be overridden using the `MAX_JIT_MOP_FLAGS_OWN_CLEAR` and `MAX_JIT_MOP_FLAGS_OWN_NOTIFY` flags, respectively, when calling the `max_jit_classex_mop_wrap()` function. Object registration and notification is covered in detail in a future chapter, but the jit.notify notify method is provided as an example.

```
// s is the servername, msg is the message, ob is the server object pointer,
// and data is extra data the server might provide for a given message
void max_jit_notify_notify(
    t_max_jit_notify *x, t_symbol *s, t_symbol *msg, void *ob, void *data)
{
    if (msg==gensym("splat")) {
        post("notify: server=%s message=%s",s->s_name,msg->s_name);
        if (!data) {
            error("splat message NULL pointer");
            return;
        }
        // here's where we output using the rightmost outlet
        // we just happen to know that "data" points to a t_atom[3]
        max_jit_obex_dumpout(x,msg,3,(t_atom *)data);
    } else {
        // pass on to the default Max MOP notification method
        max_jit_mop_notify(x,s,msg);
    }
}
```

## 23.19 Overriding the adapt and outputmode Attributes

By default, adapt and outputmode attributes are added to the MOP Max Wrapper. These attributes determine whether or not to adapt to incoming matrix attributes, and whether or not the output should calculate a new output matrix, output the last calculated matrix (freeze), pass on the input matrix (bypass). To prevent the addition of the default attributes for adapt and outputmode, you can use the `MAX_JIT_MOP_FLAGS_OWN_ADAPT`, and `MAX_JIT_MOP_FLAGS_OWN_OUTPUTMODE` flags when calling the `max_jit_classex_mop_wrap()` function. To define your own attributes, you would follow the same means of defining any attributes for a Max wrapper class with the appropriate attribute name you wish to override.

## 23.20 Defining an mproc Method

For many types of operations, it's not required to fully override the default `jit_matrix` method and any adaptation. If your object simply needs to override the way in which the Jitter class' `matrix_calc` method and outlet functions are called, you can do so by defining an `mproc` method, which will be called instead of the default behavior. The jit.3m SDK project is an example where after it calls the Jitter class' `matrix_calc` method, it queries the Jitter class' attributes and outputs max messages rather than the default `jit_matrix` message output.

```
void max_jit_3m_mproc(t_max_jit_3m **x, void *mop)
{
    t_jit_err err;
    // call internal Jitter object's matrix_calc method
    if (err=(t_jit_err) jit_object_method(
        max_jit_obex_jitob_get(x),
        _jit_sym_matrix_calc,
        jit_object_method(mop,_jit_sym_getinputlist),
        jit_object_method(mop,_jit_sym_getoutputlist)))
    {
        // report error if present
        jit_error_code(x,err);
    } else {
        // query Jitter class and makes outlet calls
        max_jit_3m_bang(x);
    }
}
```

## 23.21 The Max Class Constructor/Destructor

As we discussed in the Matrix Operator Quick Start, inside your Max class' constructor you need to allocate the matrices necessary for the MOP inputs and outputs, the corresponding matrix inlets and outlets, process matrix arguments and other MOP setup. And in your destructor, you need to free up MOP resources. Typically you would accomplish this all with the standard `max_jit_mop_setup_simple()` and `max_jit_mop_free()` functions, however there are some instances where you may need to introduce custom behavior.

### 23.21.1 Variable Inputs/Outputs

The `max_jit_mop_setup_simple()` function calls `max_jit_mop_inputs()` and `max_jit_mop_outputs()` to define any necessary proxy inlets, outlets, and internal matrices. The listing for these functions are provided below to illustrate the default behavior, and a few SDK projects we recommend investigating further are `jit.scissors`, `jit.glue`, `jit.pack`, and `jit.unpack`.

```
t_jit_err max_jit_mop_inputs(void **x)
{
    void *mop,*p,*m;
    long i,incount;
    t_jit_matrix_info info;
    t_symbol *name;
```

```

// look up object's MOP adornment
if (x&&(mop=max_jit_obex_adornment_get(x,_jit_sym_jit_mop)))
{
    incount = jit_attr_getlong(mop,_jit_sym_inputcount);
    // add proxy inlet and internal matrix for
    // all inputs except leftmost inlet
    for (i=2;i<=incount;i++) {
        max_jit_proxy_new(x,(incount+1)-i); // right to left
        if (p=jit_object_method(mop,_jit_sym_getinput,i)) {
            jit_matrix_info_default(&info);
            max_jit_mop_restrict_info(x,p,&info);
            name = jit_symbol_unique();
            m = jit_object_new(_jit_sym_jit_matrix,&info);
            m = jit_object_register(m,name);
            jit_attr_setsym(p,_jit_sym_matrixname,name);
            jit_object_method(p,_jit_sym_matrix,m);
            jit_object_attach(name, x);
        }
    }
    return JIT_ERR_NONE;
}
return JIT_ERR_INVALID_PTR;
}

t_jit_err max_jit_mop_outputs(void *x)
{
    void *mop,*p,*m;
    long i,outcount;
    t_jit_matrix_info info;
    t_symbol *name;
    if (x&&(mop=max_jit_obex_adornment_get(x,_jit_sym_jit_mop)))
    {
        outcount = jit_attr_getlong(mop,_jit_sym_outputcount);
        // add outlet and internal matrix for all outputs
        for (i=1;i<=outcount;i++) {
            max_jit_mop_matrixout_new(x,(outcount)-i); // right to left
            if (p=jit_object_method(mop,_jit_sym_getoutput,i)) {
                jit_matrix_info_default(&info);
                max_jit_mop_restrict_info(x,p,&info);
                name = jit_symbol_unique();
                m = jit_object_new(_jit_sym_jit_matrix,&info);
                m = jit_object_register(m,name);
                jit_attr_setsym(p,_jit_sym_matrixname,name);
                jit_object_method(p,_jit_sym_matrix,m);
                jit_object_attach(name, x);
            }
        }
        return JIT_ERR_NONE;
    }
    return JIT_ERR_INVALID_PTR;
}

```

## 23.21.2 Matrix Arguments

The `max_jit_mop_setup_simple()` function calls `max_jit_mop_matrix_args()` to read any matrix arguments, and if present send them to any linked inputs/outputs and disable the adapt attribute. The listing is provided below to illustrate the default behavior.

```

t_jit_err max_jit_mop_matrix_args(void *x, long argc, t_atom *argv)
{
    void *mop,*p,*m;
    long incount,outcount,attrrstart,i,j;
    t_jit_matrix_info info,info2;
    if (!(mop=max_jit_obex_adornment_get(x,_jit_sym_jit_mop)))
        return JIT_ERR_GENERIC;
    incount = jit_attr_getlong(mop,_jit_sym_inputcount);
    outcount = jit_attr_getlong(mop,_jit_sym_outputcount);
    jit_matrix_info_default(&info);
    attrrstart = max_jit_attr_args_offset(argc,argv);
    if (attrrstart&&argv) {
        jit_atom_arg_getlong(&info.planecount, 0, attrrstart, argv);
        jit_atom_arg_getsym(&info.type, 1, attrrstart, argv);
        i=2; j=0;
        while (i<attrrstart) { //dimensions
            jit_atom_arg_getlong(&(info.dim[j]), i, attrrstart, argv);
            i++; j++;
        }
    }
}
```

```

    if (j) info.dimcount=j;
    jit_attr_setlong(mop,_jit_sym_adapt,0); //adapt off
}
jit_attr_setlong(mop,_jit_sym_outputmode,1);
for (i=2;i<=incount;i++) {
    if ((p= jit_object_method(mop,_jit_sym_getinput,i)) &&
        (m= jit_object_method(p,_jit_sym_getmatrix)))
    {
        jit_object_method(m,_jit_sym_getinfo,&info2);
        if (jit_attr_getlong(p,_jit_sym_typealink)) {
            info2.type = info.type;
        }
        if (jit_attr_getlong(p,_jit_sym_planelink)) {
            info2.planecount = info.planecount;
        }
        if (jit_attr_getlong(p,_jit_sym_dimlink)) {
            info2.dimcount = info.dimcount;
            for (j=0;j<info2.dimcount;j++) {
                info2.dim[j] = info.dim[j];
            }
        }
        max_jit_mop_restrict_info(x,p,&info2);
        jit_object_method(m,_jit_sym_setinfo,&info2);
    }
}
for (i=1;i<=outcount;i++) {
    if ((p= jit_object_method(mop,_jit_sym_getoutput,i)) &&
        (m= jit_object_method(p,_jit_sym_getmatrix)))
    {
        jit_object_method(m,_jit_sym_getinfo,&info2);
        if (jit_attr_getlong(p,_jit_sym_typealink)) {
            info2.type = info.type;
        }
        if (jit_attr_getlong(p,_jit_sym_planelink)) {
            info2.planecount = info.planecount;
        }
        if (jit_attr_getlong(p,_jit_sym_dimlink)) {
            info2.dimcount = info.dimcount;
            for (j=0;j<info2.dimcount;j++) {
                info2.dim[j] = info.dim[j];
            }
        }
        max_jit_mop_restrict_info(x,p,&info2);
        jit_object_method(m,_jit_sym_setinfo,&info2);
    }
}
return JIT_ERR_NONE;
}

```

## Chapter 24

# OB3D QuickStart

The purpose of this chapter is to give a quick and high level overview of how to develop a simple Jitter OpenGL object which draws geometry within a named rendering context - we refer to such an object as an OB3D.

For this task, we will use the jit.gl.simple SDK example. More details such as how to make an OpenGL object which deals with resources such as display lists and textures, wishes to support matrix input/output, or needs greater access to OpenGL state will appear in the following chapter. This chapter assumes familiarity with Jitter's OpenGL object suite used from the Max patcher, as discussed in the Jitter Tutorial, and the preceding chapters on the Jitter object model and Max wrapper classes.

### 24.1 Defining the OB3D Jitter Class

Jitter OB3Ds typically are defined to have all or most of the common OB3D attributes and methods discussed in the Group-OB3D section of the Jitter HTML object reference. These include attributes and methods to set the rendering destination name, object name, color, lighting, texturing, modelview transform, depth buffering, polygon mode, and several other common tasks. These common attributes and methods are added by the call to the `jit_ob3d_setup()` function in your Jitter class definition, after calling `jit_class_new`, but typically prior to defining other methods and attributes. For an OB3D, Jitter needs to store additional information in your object. This information is stored in an opaque pointer in your object struct, typically named `ob3d`. The byte offset to your OB3D data pointer is passed into `jit_ob3d_setup()`. You can override any default attributes and methods added by `jit_ob3d_setup()` with the following flags:

```
#define JIT_OB3D_NO_ROTATION_SCALE      1 << 0
#define JIT_OB3D_NO_POLY_VARS           1 << 1
#define JIT_OB3D_NO_BLEND              1 << 2
#define JIT_OB3D_NO_TEXTURE            1 << 3
#define JIT_OB3D_NO_MATRIXOUTPUT       1 << 4
#define JIT_OB3D_AUTO_ONLY             1 << 5
#define JIT_OB3D_DOES_UI               1 << 6
#define JIT_OB3D_NO_DEPTH              1 << 7
#define JIT_OB3D_NO_ANTIALIAS          1 << 8
#define JIT_OB3D_NO_FOG                1 << 9
#define JIT_OB3D_NO_LIGHTING_MATERIAL  1 << 10
#define JIT_OB3D_HAS_LIGHTS            1 << 11
#define JIT_OB3D_HAS_CAMERA             1 << 12
#define JIT_OB3D_IS_RENDERER           1 << 13
#define JIT_OB3D_NO_COLOR              1 << 14
```

Aside from the attributes and methods added to your class by `jit_ob3d_setup()`, you need to define a private, untyped method bound to the symbol `ob3d_draw`. This method is where your object does all its drawing. It is called by the standard OB3D draw and drawraw methods. The OB3D draw method sets up all of the OpenGL state associated with the common OB3D attributes before calling your private `ob3d_draw` method. The drawraw method simply sets the context

before calling your private ob3d\_draw method. Because OB3Ds support being named for use within jit.gl.sketch\*'s drawobject command, you must also add a private, untyped "register" method associated with the [jit\\_object\\_register\(\)](#) function. Let's examine the \*jit.gl.simple SDK project as an example:

```
t_jit_err jit_gl_simple_init(void)
{
    long ob3d_flags = JIT_OB3D_NO_MATRIXOUTPUT; // no matrix output
    void *ob3d;
    _jit_gl_simple_class = jit_class_new("jit_gl_simple",
        (method) jit_gl_simple_new, (method) jit_gl_simple_free,
        sizeof(t_jit_gl_simple), 0L);
    // set up object extension for 3d object, customized with flags
    ob3d = jit_ob3d_setup(_jit_gl_simple_class,
        calcoffset(t_jit_gl_simple, ob3d),
        ob3d_flags);
    // define the OB3D draw method. called in automatic mode by
    // jit.gl.render or otherwise through ob3d when banged. this
    // method is A_CANT because our draw setup needs to happen
    // in the ob3d beforehand to initialize OpenGL state
    jit_class_addmethod(_jit_gl_simple_class,
        (method) jit_gl_simple_draw, "ob3d_draw", A_CANT, 0L);
    // define the dest_closing and dest_changed methods.
    // these methods are called by jit.gl.render when the
    // destination context closes or changes: for example, when
    // the user moves the window from one monitor to another. Any
    // resources your object keeps in the OpenGL machine
    // (e.g. textures, display lists, vertex shaders, etc.)
    // will need to be freed when closing, and rebuilt when it has
    // changed. In this object, these functions do nothing, and
    // could be omitted.
    jit_class_addmethod(_jit_gl_simple_class,
        (method) jit_gl_simple_dest_closing, "dest_closing", A_CANT, 0L);
    jit_class_addmethod(_jit_gl_simple_class,
        (method) jit_gl_simple_dest_changed, "dest_changed", A_CANT, 0L);
    // must register for ob3d use
    jit_class_addmethod(_jit_gl_simple_class,
        (method) jit_object_register, "register", A_CANT, 0L);
    jit_class_register(_jit_gl_simple_class);
    return JIT_ERR_NONE;
}
```

## 24.2 The Jitter Class Constructor/Destructor

In your OB3D Jitter Class constructor, you need to pass in your rendering destination name as the first argument. You should call the [jit\\_ob3d\\_new\(\)](#) function with your destination name argument to initialize the OB3D data pointer, associating it with your rendering destination. In your destructor, you need to free your OB3D data pointer with [jit←ob3d\\_free\(\)](#). The jit.gl.simple constructor and destructors are below as an example.

```
t_jit_gl_simple *jit_gl_simple_new(t_symbol *dest_name)
{
    t_jit_gl_simple *x;
    // make jit object
    if (x = (t_jit_gl_simple *) jit_object_alloc(_jit_gl_simple_class))
    {
        // create and attach ob3d
        jit_ob3d_new(x, dest_name);
    }
    else
    {
        x = NULL;
    }
    return x;
}
void jit_gl_simple_free(t_jit_gl_simple *x)
{
    // free ob3d data
    jit_ob3d_free(x);
}
```

## 24.3 The OB3D draw Method

Your OB3D draw method, bound to the ob3d\_draw symbol, is where all of your drawing code takes place. It is called automatically when your associated jit.gl.render object receives a bang, if your automatic and enabled attributes are turned on, as they are by default. It is also called if your Max wrapper object receives a bang, or the draw or drawraw messages. With the exception of the drawraw message, all of the standard OB3D object state is setup prior to calling your ob3d\_draw method, so you needn't setup things like the modelview transform, color, lighting properties, texture information, if your object doesn't have special needs. The following example from jit.gl.simple, just draws a simple quadrilateral.

```
t_jit_err jit_gl_simple_draw(t_jit_gl_simple **x)
{
    t_jit_err result = JIT_ERR_NONE;
    // draw our OpenGL geometry.
    glBegin(GL_QUADS);
    glVertex3f(-1,-1,0);
    glVertex3f(-1,1,0);
    glVertex3f(1,1,0);
    glVertex3f(1,-1,0);
    glEnd();
    return result;
}
```

Since this example is meant only to show a minimal object which draws geometry with standard OpenGL calls, there is no texture information or vertex normals specified. However, all standard OpenGL calls should work within the ob3d\_draw method. This example also doesn't show matrix output, as accomplished by jit\_ob3d\_draw\_chunk(), which will be discussed in the following chapter on OB3D details.

## 24.4 Defining the OB3D Max Wrapper Class

For OB3Ds, the Max wrapper class has less extra work than for MOPs. In your Max wrapper class definition, you need only add a call to the max\_ob3d\_setup() function to add your standard drawing methods, and the max\_jit\_ob3d\_assist() function as your assist method, unless you wish to define your own custom assist method. Everything else is similar to the standard technique of wrapping a Jitter Class demonstrated in the Max Wrapper Class chapter.

```
void ext_main(void *r)
{
    void *classex, *jitclass;
    // initialize Jitter class
    jit_gl_simple_init();
    // create Max class
    setup(&t_messlist **)&max_jit_gl_simple_class,
        (method)max_jit_gl_simple_new, (method)max_jit_gl_simple_free,
        (short)sizeof(t_max_jit_gl_simple), 0L, A_GIMME, 0);
    // specify a byte offset to keep additional information about our object
    classex = max_jit_classex_setup(calcoffset(t_max_jit_gl_simple, obex));
    // look up Jitter class in the class registry
    jitclass = jit_class_findbyname(gensym("jit_gl_simple"));
    // wrap Jitter class with the standard methods for Jitter objects
    max_jit_classex_standard_wrap(classex, jitclass, 0);
    // use standard ob3d assist method
    address((method)max_jit_ob3d_assist, "assist", A_CANT, 0);
    // add methods for 3d drawing
    max_ob3d_setup();
}
```

## 24.5 The Max Class Constructor/Destructor

Your Max class' constructor should be similar to the standard Max wrapper constructor, but the differences worth noting are that you should pass your first normal argument, which is the rendering destination, on to your Jitter OB3D constructor, and create a second outlet for matrix output, attached to your object's OB3D data. For your destructor, there

is nothing additional you need to do for OB3D. The jit.gl.simple Max class' constructor and destructor are provided as examples.

```
void *max_jit_gl_simple_new(t_symbol *s, long argc, t_atom *argv)
{
    t_max_jit_gl_simple *x;
    void *jit_ob;
    long attrstart;
    t_symbol *dest_name_sym = _jit_sym_nothing;
    if (x = (t_max_jit_gl_simple *) max_jit_obex_new(
        max_jit_gl_simple_class, gensym("jit_gl_simple")))
    {
        // get first normal arg, the destination name
        attrstart = max_jit_attr_args_offset(argc, argv);
        if (attrstart & argv)
        {
            jit_atom_arg_getsym(&dest_name_sym, 0, attrstart, argv);
        }
        // instantiate Jitter object with dest_name arg
        if (jit_ob = jit_object_new(
            gensym("jit_gl_simple"), dest_name_sym))
        {
            // set internal jitter object instance
            max_jit_obex_jitob_set(x, jit_ob);
            // add a general purpose outlet (rightmost)
            max_jit_obex_dumpout_set(x, outlet_new(x, NULL));
            // process attribute arguments
            max_jit_attr_args(x, argc, argv);
            // attach the jit object's ob3d to a new outlet
            // this outlet is used in matrixoutput mode
            max_jit_ob3d_attach(x, jit_ob, outlet_new(x, "jit_matrix"));
        }
        else
        {
            error("jit.gl.simple: could not allocate object");
            freeobject((t_object *)x);
            x = NULL;
        }
    }
    return (x);
}
void max_jit_gl_simple_free(t_max_jit_gl_simple *x)
{
    // lookup our internal Jitter object instance and free
    jit_object_free(max_jit_obex_jitob_get(x));
    // free resources associated with our obex entry
    max_jit_obex_free(x);
}
```

# Chapter 25

## OB3D Details

The purpose of this chapter is to fill in additional details of Jitter OpenGL, which we refer to as OB3Ds.

We will show how to disable and/or override default OB3D attributes and methods, how to support matrix input and output, and manage resources such as textures, display lists, and shaders. This chapter assumes familiarity with the OpenGL API and the OB3D Quick Start chapter. It is out of the scope of our documentation to cover the OpenGL API, so for information on the OpenGL API we recommend consulting the OpenGL Red Book and the many online tutorials.

### 25.1 Defining the OB3D Jitter Class

As covered in the OB3D Quick Start, Jitter OB3Ds have a large number of default attributes and methods, and require some specific methods to be defined. This section seeks to clarify these common attributes and methods and how to achieve custom behavior where necessary.

### 25.2 Declaring a Draw Method

All Jitter OB3Ds must define a method bound to the symbol `ob3d_draw`. This method takes no arguments in addition to the object struct, and should be defined with the private `A_CANT` type signature. The private `ob3d_draw` method will be called by the standard `draw`, and `drawraw` methods that are added to every OB3D. The `draw` method will set up OpenGL state associated with the default OB3D attributes before calling `ob3d_draw`, while the `drawraw` method will not.

### 25.3 Declaring Destination and Geometry Related Methods

It is possible for attributes of a Jitter OB3D or your render destination to change, requiring resources to be freed or rebuilt. There are three methods used to communicate to an OB3D when such events happen so that the OB3D can manage resources accordingly. They are: `dest_closing`, which informs an OB3D that the destination is being freed, and any context dependent resources such as textures, display lists, and shaders should be freed; `dest_changed`, which informs an OB3D that the destination has been rebuilt, and new resources can be allocated; and `rebuild_geometry`, which informs an OB3D of a change in texture units or some other attribute which affects `jit_gl_drawinfo_setup()` and other `t_jit_gl_drawinfo` related functions, such as `jit_gl_texcoord`, requiring geometry that uses such functions to be rebuilt. These methods take no arguments in addition to the object struct. The `dest_closing` and `dest_changed` methods should be defined with the private `A_CANT` type signature, and the `rebuild_geometry` method is typically defined as typed, but without arguments, so that users have the ability to explicitly call, if deemed necessary. The `jit.gl.gridshape` SDK project is a good example of these methods as it needs to free and allocate a display list as the render destination changes, and also makes use of `jit_gl_texcoord` to support multi-texturing, requiring geometry to be rebuilt as the number of texture units or other attributes change.

## 25.4 Declaring a Register Method

Since all Jitter OB3D objects are named to support reference by name in jit.gl.sketch, and other objects, it is necessary to add the default registration method, `jit_object_register()`. Object registration and notification are covered in detail in a future chapter.

## 25.5 Overriding Rotation and Scale Related Attributes

By default, each Jitter OB3D has rotate, rotatexyz, scale, and viewalign attributes added to the class by `jit_ob3d_setup()`, and these attributes are used in the `ob3d_draw_preamble()` function to set up OpenGL state prior to calling your object's draw method. These attributes can be disabled by using the `JIT_OB3D_NO_ROTATION_SCALE` flag. You can override these attributes by defining your own attributes of the same name, however, you will need to manage any necessary OpenGL state inside of your own draw method with the appropriate calls to `glMatrixMode`, `glTranslate`, `glRotate`, and `glScale`.

## 25.6 Overriding Color Related Attributes

By default, each Jitter OB3D has color, aux\_color, and smooth\_shading attributes added to the class by `jit_ob3d_setup()`, and these attributes are used in the `ob3d_draw_preamble` function prior to calling your object's draw method. These attributes can be disabled by using the `JIT_OB3D_NO_COLOR` flag. You can override these attributes by defining your own attributes of the same name, however, you will need to manage any necessary OpenGL state inside of your own draw method with the appropriate calls to `glColor` and `glShadeModel`.

## 25.7 Overriding Texture Related Attributes

By default, each Jitter OB3D has texture, capture, tex\_map, tex\_plane\_s, and tex\_plane\_t attributes added to the class by `jit_ob3d_setup()`, and these attributes are used in the `ob3d_draw_preamble()` function prior to calling your object's draw method. These attributes can be disabled by using the `JIT_OB3D_NO_TEXTURE` flag. You can override these attributes by defining your own attributes of the same name, however, you will need to manage any necessary OpenGL state inside of your own draw method with the appropriate calls to `glEnable`, `glTexGen`, `jit_gl_bindtexture`, `jit_gl←unbindtexture`, `jit_gl_begincapture`, and `jit_gl_endcapture`.

## 25.8 Overriding Lighting and Material Related Attributes

By default, each Jitter OB3D has lighting\_enable, auto\_material, shininess, mat\_ambient, mat\_diffuse, mat\_specular, and mat\_emission attributes added to the class by `jit_ob3d_setup()`, and these attributes are used in the `ob3d←draw_preamble` function prior to calling your object's draw method. These attributes can be disabled by using the `JIT_OB3D_NO_LIGHTING_MATERIAL` flag. You can override these attributes by defining your own attributes of the same name, however, you will need to manage any necessary OpenGL state inside of your own draw method with the appropriate calls to `glEnable`, `glLight`, `glLightModel`, and `glMaterial`.

## 25.9 Overriding Fog Related Attributes

By default, each Jitter OB3D has fog and fog\_params attributes added to the class by jit\_ob3d\_setup(), and these attributes are used in the ob3d\_draw\_preamble function prior to calling your object's draw method. These attributes can be disabled by using the [JIT\\_OB3D\\_NO\\_FOG](#) flag. You can override these attributes by defining your own attributes of the same name, however, you will need to manage any necessary OpenGL state inside of your own draw method with the appropriate calls to glEnable, glHint, and glFog.

## 25.10 Overriding Polygon Variable Related Attributes

By default, each Jitter OB3D has poly\_mode, cull\_face, point\_size, and line\_width attributes added to the class by jit\_ob3d\_setup(), and these attributes are used in the ob3d\_draw\_preamble function prior to calling your object's draw method. These attributes can be disabled by using the [JIT\\_OB3D\\_NO\\_POLY\\_VARS](#) flag. You can override these attributes by defining your own attributes of the same name, however, you will need to manage any necessary OpenGL state inside of your own draw method with the appropriate calls to glPolygonMode, glEnable, glCullFace, glPointSize, and glLineWidth.

## 25.11 Overriding Blending Related Attributes

By default, each Jitter OB3D has blend\_mode and blend\_enable attributes added to the class by jit\_ob3d\_setup(), and these attributes are used in the ob3d\_draw\_preamble function prior to calling your object's draw method. These attributes can be disabled by using the [JIT\\_OB3D\\_NO\\_BLEND](#) flag. You can override these attributes by defining your own attributes of the same name, however, you will need to manage any necessary OpenGL state inside of your own draw method with the appropriate calls to glEnable and glBlendFunc.

## 25.12 Overriding Depth Buffer and Antialiasing Related Attributes

By default, each Jitter OB3D has depth\_enable and antialias attributes added to the class by jit\_ob3d\_setup(), and these attributes are used in your ob3d\_draw\_preamble function prior to calling your object's draw method. These attributes can be disabled by using the [JIT\\_OB3D\\_NO\\_DEPTH](#) and [JIT\\_OB3D\\_NO\\_ANTIALIAS](#) flags, respectively. You can override these attributes by defining your own attributes of the same name, however, you will need to manage any necessary OpenGL state inside of your own draw method with the appropriate calls to glEnable and glHint.

## 25.13 Overriding Matrixoutput and Automatic Attributes

By default, each Jitter OB3D has matrixoutput and automatic attributes added to the class by jit\_ob3d\_setup(), and these attributes are used in the ob3d\_draw\_preamble function prior to calling your object's draw method. These attributes can be disabled by using the [JIT\\_OB3D\\_NO\\_MATRIXOUTPUT](#) and [JIT\\_OB3D\\_AUTO\\_ONLY](#) flags, respectively. You can override these attributes by defining your own attributes of the same name.

## 25.14 Declaring a User Interface Object

It is possible to declare a user interface OB3D, such as jit.gl.handle. To do so, you must use the `JIT_OB3D_DOES_UI` flag to `jit_ob3d_setup()`, and define a method bound to the symbol `ob3d_ui`, with the private `A_CANT` type signature and prototype similar to the following example from `jit.gl.handle`:

```
t_jit_err jit_gl_handle_ui(t_jit_gl_handle **x,
    t_line_3d *p_line, t_wind_mouse_info *p_mouse);
```

## 25.15 The Jitter Class Constructor and Destructor

Inside your Jitter class constructor, you must call `jit_ob3d_new()` with a pointer to your newly allocated object, and your render destination name. The `jit_ob3d_new()` function allocates an opaque structure that stores the standard OB3D attributes and some additional OB3D state, initializing them to default values, and then setting the pointer at the byte offset specified when calling the `jit_ob3d_setup()` function in your class definition. If your object supports matrix output or simply uses the `t_jit_glchunk` structure when drawing, you should typically allocate your initial `t_jit_glchunk` in your constructor using the `jit_glchunk_new()` or `jit_glchunk_grid_new()` functions. Use of the `t_jit_glchunk` structure and matrix output is described later in this chapter. Similarly, your OB3D Jitter class destructor must call `jit_ob3d_free()` to free the opaque structure used for common OB3D state, free any allocated instances of `t_jit_glchunk` with `jit_glchunk_free()`, and free any other resources allocated such as display lists or textures.

## 25.16 The OB3D Draw Method

The `ob3d_draw` method is where all the drawing in your object should take place. It is also where you should typically allocate context dependent resources or query the context state, since you know that your context is valid and has been set. For the most part, the drawing you will perform in your `ob3d_draw` method will be pure and simple OpenGL, though there are a few caveats which we will cover.

## 25.17 The `t_jit_glchunk` Structure and Matrix Output

Since Jitter is a general purpose matrix processing framework, it makes sense that you would have the ability to pass geometry information through a Jitter network as matrices if your geometry is well suited to a matrix representation. The cells of your matrix can hold vertex information such as position, texture coordinates, normal vectors, color, and edge flags, and are documented in the "Geometry Under The Hood" Jitter Tutorial. You also have the option of specifying a connections matrix to reference the connectivity of the vertices if it is not implicit in the matrix representation, and a drawing primitive to use when drawing the vertices.

All this information, and whether or not the geometry matrix should be rendered immediately or sent through the Jitter network is managed with the `t_jit_glchunk`. An SDK example which demonstrates the use of `t_jit_glchunk` is `jit.gl.gridshape`. The `t_jit_glchunk` structure along with the vertex matrix it contains is allocated by the `jit_glchunk_new()` or `jit_glchunk_grid_new()` functions, freed with the `jit_glchunk_delete()` function, and drawn with the `jit_ob3d_draw_chunk()` function. For reference, the `t_jit_glchunk` structure and relevant chunk flags are provided below:

```
// jit_glchunk is a public structure to store one
// gl-command's-worth of data, in a format which
// can be passed easily to glDrawRangeElements.
typedef struct _jit_glchunk
{
    t_symbol    *prim;           // GL_TRI_STRIP, GL_TRIANGLES, etc.
    t_jit_object *m_vertex;     // jit_matrix of xyzst... data.
    t_symbol    *m_vertex_name;  // vertex matrix name
```

```

t_jit_object *m_index;           // optional 1d connection matrix
t_symbol    *m_index_name;      // connection matrix name
unsigned long m_flags;          // special flags
void        *next_chunk;        // singly linked list, typically NULL
} t_jit_glchunk;
// flags for chunk creation
#define JIT_GL_CHUNK_IGNORE_TEXTURES     1 << 0
#define JIT_GL_CHUNK_IGNORE_NORMALS    1 << 1
#define JIT_GL_CHUNK_IGNORE_COLORS    1 << 2
#define JIT_GL_CHUNK_IGNORE_EDGES     1 << 3

```

## 25.18 OB3D OpenGL Caveats

While you can use any standard OpenGL calls inside of your ob3d\_draw method. There are a few things worth noting to follow Jitter conventions. The first of which is the binding of texture coordinates. Since Jitter OB3Ds support multi-texturing by default, it is not necessarily satisfactory to submit only one texture coordinate with glTexCoord. Jitter provides some utility routines to set the texture coordinates for as many texture units which are bound, jit\_gl→\_texcoord(1/2/3)(f/fv). Determining how many texture units have been bound by the default OB3D attributes requires some overhead, so rather than perform this overhead with every jit\_gl\_texcoord call, the jit\_gl\_texcoord functions take a `t_jit_gl_drawinfo` struct as an argument. This struct can be setup once before rendering many vertices with the jit→\_gl\_drawinfo\_setup function. Example use of jit\_gl\_texcoord and jit\_gl\_drawinfo\_setup is in the jit.gl.videoplane SDK project. Another Jitter specific mechanism is the means to bind textures using named instances of jit.gl.texture. It is possible to create and bind your own textures in an OB3D, but you must then perform all maintenance instead of relying on jit.gl.texture to handle this work for you. To bind and unbind an instance of jit.gl.texture, you should call the jit→\_gl\_bindtexture and jit\_gl\_unbindtexture functions, which take a `t_jit_gl_drawinfo` argument, a symbol with the name of the jit.gl.texture instance, and an integer for which texture unit to bind. Unlike binding ordinary textures in OpenGL, it is important to unbind instances of jit.gl.texture, or else problems may arise.

## 25.19 Getting Information About the OB3D Attributes

Though the default OB3D attributes are typically relevant to the code which is automatically handled for your object prior to calling the ob3d\_draw method, it is sometimes necessary to access these values. Since the default OB3D attributes are stored in an opaque ob3d struct member, they are not accessible by your object with a simple struct pointer dereference. Instead, you need to use the jit\_attr\_get\* functions to access these attributes. You should pass in your object struct as the first argument to these functions rather than your ob3d struct member. For example:

```

float pos[3];
jit_attr_getfloat_array(x, gensym("position"), 3, pos);

```

Note that if you are acquiring this value often, it is preferable to generate the symbol in advance rather than generate the symbol for every call.

## 25.20 Getting Information About the Context

From within the ob3d\_draw, dest\_closing, and dest\_changed methods, the rendering context has always been set, and you can get a handle to the native context using either the aglGetCurrentContext or wglGetCurrentContext functions. One can also in these methods use standard OpenGL glGet\* functions to determine the context's OpenGL state, such as the viewport, transformation matrix. It is not recommended to try and acquire the native context from other methods, or query the OpenGL state as it may not be valid.

## 25.21 Playing Well with Others

It is important to recognize that OpenGL state is persistent, and that there may be objects which rely on OpenGL state that are drawn after your object draws itself. If your object makes any changes to OpenGL state that might affect objects that follow, you should restore the OpenGL state to whatever it was before your routine was called. For example, if your object changes the texture transformation matrix, you should push and pop the texture transformation matrix with `glMatrixMode`, `glPushMatrix`, and `glPopMatrix`, to prevent any problems with other objects.

## 25.22 Defining the OB3D Max Wrapper Class

As mentioned in the OB3D Quick Start, in your Max wrapper class definition, you need only add a call to the `max←_ob3d_setup()` function to add your standard drawing methods, and the `max_jit_ob3d_assist()` function as your assist method, unless you wish to define your own custom assist method. Everything else is similar to the standard technique of wrapping a Jitter Class demonstrated in the Max Wrapper Class chapter. Please consult the OB3D Quick Start chapter and the `jit.gl.simple` SDK project for all necessary information related to the OB3D Max wrapper class.

## 25.23 Matrix Input

Sometimes it is desirable for an OB3D also support incoming matrices as is the case with `jit.gl.videoplane` or `jit.gl.mesh`. It is not recommended to mix and match OB3Ds with MOPs. Conflicts arise with respect to arguments, standard inlets and outlets. Instead, if you wish to support matrix input in your OB3D, you should simply add to your Jitter class a method bound to the symbol `jit_matrix`, and handle the incoming matrix data according to your needs - for example as texture data in the case of `jit.gl.videoplane`, or geometry data in the case of `jit.gl.mesh`. The `jit.gl.videoplane` SDK project provides an example of an OB3D which also supports matrix input. When it is necessary to have multiple input matrices, this is typically managed by either declaring alternately named methods for each input, or exposing an attribute that specifies which input the `jit_matrix` method assumes it is being called with. Note that this requires additional logic within the Max wrapper class to map to inlets, as it is not handled automatically.

## Chapter 26

# Scheduler and Low Priority Queue Issues

In Max, there are a few threads of execution.

The details of these threads are highlighted in the Max documentation and the article, "Event Priority in Max (Scheduler vs. Queue)". In this chapter, we won't cover all these details and restrict our discussion to the scheduler (which when overdrive is on runs in a separate and high priority thread) and the low priority queue (which always runs in the main application thread). As far as Jitter is concerned, we won't consider the real time audio thread or the case of scheduler in audio interrupt, where the scheduler runs in this real time audio thread.

By default, Jitter performs all drawing and matrix processing in the main application thread, with events serviced from the low priority queue. The reason for this low priority processing is to prevent high timing events such as note triggering or audio DSP from suffering timing problems due to visual processing. Jitter also exploits the low priority queue as a mechanism for graceful temporal downsampling of the visual stream in the instance that the processing requested is too demanding to be calculated in real-time. This results in dropped frames in the output when the demands can't be met. With audio, it's not sufficient to just drop frames of samples, since there will be an audible click, but with images, the last image will persist if a new one isn't generated at some fixed sampling rate.

### 26.1 Defer and Usurp

The mechanisms which enforce execution of Jitter drawing and matrix processing from within the low priority queue we will call "defer" and "usurp". The defer mechanism will take any high priority events and create a corresponding low priority event at the end of the low priority queue. The defer mechanism ensures that the events will not be executed from the high priority scheduler thread, but does not prevent scheduler backlog with the temporal downsampling mentioned above. To accomplish this, the usurp mechanism must be used. The usurp mechanism will use no more than one low priority queue element for the task requested (either a method call or attribute setter). The way usurp works is that if there is no pending event for the method or attribute call, a new event is placed at the end of the low priority queue. If there is already an event pending, the usurp mechanism will not place a new event on the end of the low priority queue, but rather "usurp" the arguments for the event waiting to be passed to the method or attribute call. This way, if a high priority metronome is rapidly sending values to set an attribute, while the initial low priority event is waiting to be processed, the value to be set is constantly being updated ("usurped") and only the value at the time of servicing the event will be used.

It is important to note that the defer and usurp mechanisms only work as called from within the Max patcher. For any methods which are called from a text based programming language, such as C, Java, or JavaScript, the defer and usurp mechanisms are bypassed. This may be something you need to pay attention to and handle yourself if you are making such calls from a text based programming language and need the defer or usurp behavior.

## 26.2 Using Defer and Usurp in Jitter Object Methods

When defining a method in Jitter, there is the possibility to define a type signature for the method just as one would do in Max. Typical type signatures include typical atom elements such as `A_LONG`, `A_FLOAT`, and `A_SYM`; or the corresponding default value versions `A_DEFLONG`, `A_DEFFLOAT`, `A_DEFSYM`; or the variable argument version `A_GIMME` which provides a list of atoms and the number of atoms provided; or the private and untyped status of `A_CANT` used for methods which are not exposed to the patcher and require additional C function prototype information in order to call. While these type signatures can be used within Jitter objects, most methods exposed to the patcher interface make use of either the defer or usurp mechanism as defined by two new type signatures `A_DEFER_LOW` or `A_USURP_LOW`. Methods defined with the `A_DEFER_LOW`, or `A_USURP_LOW` type signatures should conform to the same variable argument prototype as `A_GIMME` methods, but behind the scenes, Jitter will make use of the defer and usurp mechanism to enforce the appropriate behavior.

An example of two methods from `jit.gl.videoplane` which use these mechanisms is below:

```
// add a usurping jit_matrix method
jit_class_addmethod(_jit_gl_videoplane_class, (method) jit_gl_videoplane_jit_matrix, "jit_matrix", A_USURP_LOW,
0);
// add a deferred sendtexture method
jit_class_addmethod(_jit_gl_videoplane_class, (method) jit_gl_videoplane_sendtexture, "sendtexture",
A_DEFER_LOW, 0);
The implementation of these methods is below:
void jit_gl_videoplane_jit_matrix(t_jit_gl_videoplane **x, t_symbol *s, int argc, t_atom *argv)
{
    t_symbol *name;
    void *m;
    t_jit_matrix_info info;
    long dim[2];
    if ((name=atom_getsym(argv)) != _jit_sym_nothing) {
        m = jit_object_findregistered(name);
        if (!m) {
            error("jit.gl.videoplane: couldn't get matrix object!");
            return;
        }
    }
    if (x->texture) {
        jit_object_method(m, _jit_sym_getinfo, &info);
        jit_attr_getlong_array(x->texture, _jit_sym_dim, 2, dim);
        jit_object_method(x->texture, s, s, argc, argv);
        jit_attr_setsym(x, ps_texture, x->texturename);
    }
}
void jit_gl_videoplane_sendtexture(t_jit_gl_videoplane **x, t_symbol *s, int argc, t_atom *argv)
{
    if (x->texture) {
        s = atom_getsym(argv);
        argc--;
        if (argc)
            argv++;
        else
            argv = NULL;
        object_method_typed(x->texture, s, argc, argv, NULL);
    }
}
```

From inspecting the header files, you may note that there are also `A_DEFER` and `A_USURP` type signatures, but these should be considered obsolete, as they make use of the problematic deferral strategy of placing the event at the front of the low priority queue and have the potential of reversing message sequencing.

## 26.3 Using Defer and Usurp in Jitter Object Attributes

Unlike methods, attributes do not make use of type signatures for their getter and setter accessor methods. Instead they should always be prototyped similar to `A_GIMME`, but with an attribute object being passed in place of the traditional method symbol pointer of the `A_GIMME` signature. So the way you can specify to use the defer and usurp mechanisms for attribute accessors are through the attribute flags argument to the attribute constructor. For the getter accessor

method, you can use `JIT_ATTR_GET_DEFER_LOW` or `JIT_ATTR_GET_USURP_LOW` flags. For the setter accessor method, you can use `JIT_ATTR_SET_DEFER_LOW` or `JIT_ATTR_SET_USURP_LOW` flags.

An example attribute definition from `jit.gl.videoplane` is below:

```
attrflags = JIT_ATTR_GET_DEFER_LOW | JIT_ATTR_SET_USURP_LOW;
attr = jit_object_new(_jit_sym_jit_attr_offset, "displaylist", _jit_sym_char, attrflags,
    (method) 0L, (method) jit_gl_videoplane_displaylist, calcoffset(t_jit_gl_videoplane, displaylist));
jit_class_addattr(_jit_gl_videoplane_class, attr);
```

You may have noticed that like previous code example, all Jitter object attributes which are not private have been defined with getter accessors which use the defer mechanism (`JIT_ATTR_GET_DEFER_LOW`) and setter accessors which use the usurp mechanism (`JIT_ATTR_SET_USURP_LOW`). This is the recommended style of exposing Jitter object attributes to the patcher, since there are many cases where at high priority an attribute is set repeatedly and we want both the latest high priority value when the next calculation is made at low priority and no low priority queue backlog from generating more events at high priority than can be processed at low priority. The defer mechanism is used for getter accessor methods so that every attribute query results in a corresponding output message out the dump outlet. Otherwise certain patcher logic could easily become confused. If a different behavior is required by the Max programmer, they can make use of the `jit.qball` object to force either the defer or usurp mechanisms to be used for their message stream.

## 26.4 Using Defer and Usurp in the Max Wrapper Object

Most of the above is also true when declaring methods and attributes in the Max wrapper object, however the function calls which are used are slightly different. You must use the special max object function calls `max_addmethod_defer_low()` and `max_addmethod_usurp_low()` for methods, and `max_jit_classex_addattr()` for attributes. Below are examples from `jit.matrixset`. Note that there is no type signature provided for either `max_addmethod_defer_low()` or `max_addmethod_usurp_low()`.

```
// add a deferred "exportmovie" method
max_addmethod_defer_low((method)max_jit_matrixset_export_movie, "exportmovie");
// add a usurped outputmatrix method
max_addmethod_usurp_low((method)max_jit_matrixset_outputmatrix, "outputmatrix");
// add index attribute
attrflags = JIT_ATTR_GET_DEFER_LOW | JIT_ATTR_SET_USURP_LOW ;
attr = jit_object_new(_jit_sym_jit_attr_offset, "index", _jit_sym_long, attrflags,
    (method) 0L, (method) 0L, calcoffset(t_max_jit_matrixset, index));
max_jit_classex_addattr(p, attr);
```

## 26.5 When Not to Use the Usurp Mechanism

The bang method for Jitter MOP objects uses the usurp mechanism to drop frames when the number of bang messages cannot be handled in real time. However, `jit.gl.render`'s bang method does not behave this way, and instead uses the defer mechanism. At first this might seem counterintuitive, however, because rendering in OpenGL with `jit.gl.render` uses a group of messages to perform erasing, any non automatic drawing of objects, and then a drawing of automatic clients and a swap to the screen with the bang method, it is not an atomic action (i.e. requires a sequence of different events rather than a single event). Since the usurp mechanism is method or attribute specific with regard to the events which are being usurped, it only works for atomic actions. For this reason, it is important for users to perform some drop framing behavior before triggering the message sequence, typically accomplished with `qmetro` or `jit.qball`. If your object has some operation which requires a sequence of events in a similar fashion as `jit.gl.render`, then it would be best to use the defer mechanism rather than the usurp mechanism for relevant methods.

## 26.6 Overriding Defer and Usurp

There are instances where the user does not wish to be limited to processing Jitter matrices at low priority, such as when Jitter matrices are used for tasks other than realtime image processing—for example, parameter interpolation or matrices containing audio data. For these tasks, the jit.qfaker object is provided for advanced users which are aware of the potential problems involved in bypassing these low priority mechanisms. As mentioned above, when programming in a text based language, these mechanisms aren't used and all method and attribute accessor calls are synchronous. Therefore there typically isn't a need to consider overriding this behavior from a text based language. However, for certain externals which wish to simulate the jit.qfaker behavior, we expose the max\_jit\_queuestate() function to override Jitter's detection of queue state for the defer and usurp mechanisms. It is also possible to query what jitter believes the queue state to be with the max\_jit\_getqueuestate() function. This is the function employed by the defer and usurp mechanisms. The source code for these functions is below for reference.

```
long max_jit_queuestate(long state)
{
    long rv=_max_jit_queuestate;
    _max_jit_queuestate = (state!=0);
    return rv;
}
long max_jit_getqueuestate(void)
{
    // always return true if faking
    if (_max_jit_queuestate) return 1;
    return !sched_isinpoll();
}
```

## Chapter 27

# Jitter Object Registration and Notification

In Jitter, matrices are passed around as named references between Max objects.

This named reference is created since Jitter registers these matrices with the corresponding name using the `jit_object_register()` function. Object registration is useful for a few reasons. First, registered matrices can be resolved by name using the `jit_object_findregistered()` function. Secondly, registered objects can send event notification to clients who have attached to them using `jit_object_attach()`. Lastly, under certain circumstances, the object registration process can be used to have multiple external references to a single instance of an object as is the case with `jit.matrix`.

### 27.1 Registering Named Objects

To register an object, one can use the `jit_object_register()` function, which is equivalent to the Max `object_register()` function in the namespace associated with `gensym("jitter")`. Traditionally in Jitter, we bind `jit_object_register()` to the "register" method for an object and use `jit_object_method()` to call this method. For example, from the `jit.notify` SDK example:

```
// allocate the Jitter object
if (o= jit_object_new(gensym("jit_notify"))) {
    ...
    // generate a unique name
    x->servername = jit_symbol_unique();
    // register the object with the given name
    jit_object_method(o, _jit_sym_register, x->servername);
    ...
}
```

If not using a specific name, it is good to use the `jit_symbol_unique()` function as above to generate a unique name which is slated for re-use once a registered object is freed. This prevents excess memory usage by the symbol table as associated with these unique names.

If you wish the object to have multiple references to a single instance with some name, as is common with the `jit.matrix` object, it is essential to use the return value of `jit_object_register()` in any instance where the object pointer is saved after registration. This is because if the registered object with the same class already exists, the object attempting to be registered will be freed, and the already registered object of the same class will be returned, its reference count having been incremented. This is not typically an issue outside of registering `jit.matrix` objects, although you may have a need for this type of implementation in other situations. Most other situations in which object registration is used within Jitter only expects and/or permits a single instance to be registered. In the above example, we know that this is safe to do, as we are using `jit_symbol_unique()` to generate a unique name.

It is also possible to unregister named objects, with the `jit_object_unregister()` function, but typically this is handled for you when your object is freed, or if your object is registered again with a different name. This is not often used in the Jitter code base except within these contexts.

## 27.2 Looking Up an Object by Name

Registered objects can be found by name using the [jit\\_object\\_findregistered\(\)](#) function. For example named matrices are resolved using this function. Most Matrix Operator objects have this done for them by the default MOP code, but for example any MOP which has its own jit\_matrix method, such as the jit.pack SDK example will make use of [jit\\_object\\_findregistered\(\)](#) inside its jit\_matrix method:

```
// get our matrix name from the atom arguments provided
matrixname = jit_atom_getsym(argv);
// look up based on name
matrix = jit_object_findregistered(matrixname);
// make sure that it is a valid pointer and has a "class_jit_matrix" method which returns 1
if (matrix&&jit_object_method(matrix, _jit_sym_class_jit_matrix)) {
    ...
}
```

## 27.3 Attaching to Named Objects

Once an object has been registered, it can be considered a server to which clients attach to be notified of various events. To attach to a named object, use the the [jit\\_object\\_attach\(\)](#) function. Similarly to detach from a named object, use the [jit\\_object\\_detach\(\)](#) function. It is typical to detach from a server in your object's destructor, or any time your object is switching which server it is attached to. For your client object to receive any notification from the server object, it is important for your object to have defined a "notify" method which will receive the notification from all objects it is attached to.

Below is the jit.notify SDK example's max wrapper object's notify method, which receives some atom values from its internal Jitter object instance. Since this object is a Matrix Operator, it is important in the following example that jit.notify calls the [max\\_jit\\_classex\\_mop\\_wrap\(\)](#) function with the [MAX\\_JIT\\_MOP\\_FLAGS\\_OWN\\_NOTIFY](#) flag to override the default MOP notify method, and that we pass on all other messages to the standard [max\\_jit\\_mop\\_notify\(\)](#) method so that the default MOP code is informed of any changes to the input and output matrices.

```
// s is the servername, msg is the message, ob is the server object pointer,
// and data is extra data the server might provide for a given message
void max_jit_notify_notify(t_max_jit_notify *x, t_symbol *s, t_symbol *msg, void *ob, void *data)
{
    if (msg==gensym("splat")) {
        post("notify: server=%s message=%s",s->s_name,msg->s_name);
        if (!data) {
            error("splat message NULL pointer");
            return;
        }
        // here's where we output using the rightmost outlet
        // we just happen to know that "data" points to a t_atom[3]
        // alternately you could use max_jit_obex_dumpout_get just to get
        // the outlet pointer
        max_jit_obex_dumpout(x,msg,3,(t_atom *)data);
    } else {
        // since we are a MOP, we are also attached to all the matrices for each input/output
        // so we need to deal with this by calling the default mop notify method
        // (this is how mops handle their matrices getting new names/freed/modified)
        max_jit_mop_notify(x,s,msg);
    }
}
```

## 27.4 Notifying Clients

If you are making an object which is to be registered, and wish to send custom notification to clients in addition to the default notification that attributes send to all clients when the attribute is modified, and the default object free notification, then you will want to use the [jit\\_object\\_notify\(\)](#) function. This function lets you determine a message name to use for notification and optionally specify additional, but untyped data to all clients. If you choose to send additional data to

clients, it is necessary for all client code to know how to unpack this information. Below is the example from the jit.notify SDK example which uses the notification mechanism to send some data to its max wrapper object:

```
t_atom foo[3];
jit_atom_setlong(&foo[0],1);
jit_atom_setlong(&foo[1],2);
jit_atom_setlong(&foo[2],3);
jit_object_notify(x,gensym("splat"), foo);
```



## Chapter 28

# Using Jitter Objects in C

When developing for Jitter in C, the functionality of pre-existing Jitter objects can be used.

In this chapter, we'll briefly examine instantiation and incorporation of the features of the jit.movie and jit.qt.record objects from your C code.

### 28.1 Example 1: the t\_jit\_qt\_movie object

Using an object like t\_jit\_qt\_movie from your own code is fairly straightforward. Since it's a standard Jitter object, we can use jit\_object\_new() and jit\_object\_free() for instantiation and freeing, jit\_object\_method() for sending messages, and jit\_attr\_get... and jit\_attr\_set... for getting and setting attributes.

For instance, in the following code snippet, we'll create a t\_jit\_qt\_movie object, read a pre-specified movie from disk, and decompress its first frame into a matrix, set to the native size of the movie.

```
void jit_foo_read_first_movie_frame(
    t_jit_foo **x, t_symbol *s, long ac, t_atom *av)
{
    void *qtmovie;
    // create the t_jit_qt_movie object, sized to 1x1
    qtmovie = jit_object_new(gensym("jit_qt_movie"), 1, 1);
    if (qtmovie) {
        t_atom rv; // will contain rvarr, with any return values
                   // from our "read" call
        t_object *rvarr; // the t_atomarray with the actual
                         // return values
        // turn off autostart
        jit_attr_setlong(qtmovie, gensym("autostart"), 0);
        // read the movie, just pass in the args to our function
        object_method_typed(qtmovie, gensym("read"), ac, av, &rv);
        // check the return value & verify that the movie loaded
        if (rvarr = jit_atom_getobj(&rv)) {
            long rvac = 0;
            t_atom *rvav = NULL;
            object_getvalueof(rvarr, &rvac, &rvav);
            if (rvac && rvav) {
                // just as in Max, we get a list: "filename success";
                // success of 1 means the read was successful
                if (rvac > 1 && jit_atom_getlong(rvav + 1)) {
                    long dim[2];
                    void *matrix;
                    t_jit_matrix_info info;
                    // get our movie's native dims
                    jit_attr_getlong_array(qtmovie, gensym("movie_dim"),
                        2, dim);
                    // set the t_jit_qt_movie's dim to match
                    jit_object_method(qtmovie, _jit_sym_dim, dim[0], dim[1]);
                }
            }
        }
    }
}
```

```

// set our matrix up to match
jit_matrix_info_default(&info);
info.type = _jit_sym_char;
info.planeCount = 4;
info.dimCount = 2;
info.dim[0] = dim[0];
info.dim[1] = dim[1];
matrix = jit_object_new(_jit_sym_jit_matrix, &info);
if (matrix) {
    // call the t_jit_qt_movie's matrix_calc method
    // with our matrix as an argument
    err = (t_jit_err) jit_object_method(qtmovie,
        _jit_sym_matrix_calc, NULL, matrix);
    if (err != JIT_ERR_NONE) {
        error("something went wrong");
    }
    // do something with the matrix
    // free the matrix
    jit_object_free(matrix);
}
freebytes(rvav, sizeof(t_atom) * rvac);
freeobject(rvarr);
}
jit_object_free(qtmovie);
}
}

```

Naturally, we could also set the `t_jit_qt_movie` object's `time` attribute, or call its `or` `frame` method, to recall an arbitrary point in time. In fact, nearly every documented method and attribute of the `jit.movie` object, as it functions in the Max interface, is available from C. The exceptions are those functions implemented in the Max wrapper object, such as `framedump`.

## Chapter 29

# JXF File Specification

The Jitter File Format (JXF) stores matrix data in a binary (not human-readable) form.

When using Jitter you can create JXF files by sending the write message to a jit.matrix object. Conversely you can read JXF files from disk using the read message. This section will cover first the API functions that one can use from C to read and write JXF files. Then it will break down the file format at the bit level.

### 29.1 The Binary JXF API

Most Jitter users do not need or want to know about the internal binary format of a JXF-file. Even users who want to read and write JXF-files from C do not need to know the internal details if they use the functions of the Jitter API for the binary interface. Not only is the API more convenient, but using the functions provided by Cycling '74 may protect your code from having to be altered in the future in the event of a specification change.

There are two primary functions one should use to read data from a JXF file. `jit_bin_read_header()` reads the version number and the size of the file from the header, and has the following signature:

```
t_jit_err jit_bin_read_header(t_filehandle fh, ulong *version, long *filesize)
```

`jit_bin_read_matrix()` imports matrix data from a file to a matrix, resizing the matrix if necessary, and has the following signature:

```
t_jit_err jit_bin_read_matrix(t_filehandle fh, void *matrix)
```

Here's a chunk of code that shows how to read a matrix from disk:

```
if (!err= path_opensysfile(filename, path, &fh, READ_PERM)) {
    //all is well
} else {
    error("jit.matrix: can't open file %s", name->s_name);
    goto out;
}
if (jit_bin_read_header(fh,&version,&filesize)) {
    error("jit.matrix: improper file format %s", name->s_name);
    sysfile_close(fh);
    goto out;
}
if (jit_bin_read_matrix(fh,matrix)) {
    error("jit.matrix: improper file format %s", name->s_name);
    sysfile_close(fh);
    goto out;
}
sysfile_close(fh);
```

Similarly there are two functions one should use when writing data to a JXF file. `jit_bin_write_header()` writes a header to a file, and has the following signature:

```
t_jit_err jit_bin_write_header(t_filehandle fh, long filesize)
```

`jit_bin_write_matrix()` writes a matrix to a file, and has the following signature:

```
t_jit_err jit_bin_write_matrix(t_filehandle fh, void *matrix)
```

Here's a section of code that shows how you might write a file with one matrix. Note that the initial filesize argument to `jit_bin_write_header()` is bogus, but that the header is written again at the end of the operation when the filesize can be determined from the file position after writing the matrix.

```
if (err=path_createsysfile(filename, path, type, &fh)) {
    error("jit.matrix: could not create file %s", name->s_name);
    goto out;
}
if (jit_bin_write_header(fh,0)) {
    error("jit.matrix: could not write header %s", matrixName->s_name);
    sysfile_close(fh);
    goto out;
}
if (jit_bin_write_matrix(fh,pointerToMatrix)) {
    error("jit.matrix: could not write matrix %s", matrixName->s_name);
    sysfile_close(fh);
    goto out;
}
sysfile_getpos(fh, &position);
sysfile_seteof(fh, position);
if (jit_bin_write_header(fh,position)) {
    error("jit.matrix: could not write header %s",
matrixName->s_name);
    sysfile_close(fh);
    goto out;
}
sysfile_close(fh);
```

## 29.2 Specification of the JXF Format

The internal format of JXF-files is based on the Interchange File Format (IFF) ([http://en.wikipedia.org/wiki/Interchange\\_File\\_Format](http://en.wikipedia.org/wiki/Interchange_File_Format)). An IFF file is built up from chunks. All data in IFF files is big-endian. Several convenience macros defined in `jit.byteorder.h` are available to help convert numbers to the proper format before and after they're written to and read from a JXF file: `BE_I32()` can be called on 32-bit integers, `BE_F32()` on 32-bit floats, and `BE_F64()` on 64-bit doubles.

Each chunk in an IFF file begins with a four character Type ID. This is followed by a 32-bit unsigned integer specifying the size of the chunk content in bytes. In a JXF file, the 32-bit integer part of the first chunk tells us the size of the file, and all the subsequent chunks, which begin immediately after the first chunk, contain matrices. In the future chunks may also be used to store other kinds of data.

Here is a tabular overview of an example minimal JXF file.

Container Chunk

groupID	JIT_BIN_CHUNK_CONTAINER ('FORM')
File size	32-bit int
IFF Type	JIT_BIN_FORMAT ('JIT!')
Format Chunk	
chunkID	JIT_BIN_CHUNK_FORMAT_VERSION ('FVER')
Chunk size	12 bytes
Version	JIT_BIN_VERSION_1 (0x3C93DC80)

Matrix Chunk	
chunk ID	JIT_BIN_CHUNK_MATRIX ('MTRX')
chunk size	32-bit int
offset	32-bit int
type	4-char
planecount	32-bit int
dimcount	32-bit int
dim	Array of 32-bit ints that contain the dimensions
data	

The data offset of the matrix chunk represents the offset, in bytes, from the beginning of the chunk to the beginning of the data portion of the chunk. The type is one of CHAR, LONG, FL32 and FL64. The dim array contains dimcount elements, each of which is a 32-bit int. The data portion consists of the cells of the matrix written out one at a time in row-major order. Planar data is multiplexed in each cell. For example, a 3-plane 2 by 2 matrix would be written out in the following order:

Plane	Dim 0	Dim 1
0	0	0
1	0	0
2	0	0
0	1	0
1	1	0
2	1	0
0	0	1
1	0	1
2	0	1
0	1	1
1	1	1
2	1	1

The various chunks discussed above can be represented by the C structs listed below:

```

typedef struct _jit_bin_chunk_container
{
    ulong    ckid;          //FORM'
    long     cksize;        //filesize
    ulong    formtype;      //JIT!
} t_jit_bin_chunk_container;
typedef struct _jit_bin_chunk_format_version
{
    ulong    ckid;          //FVER'
    long     cksize;        //12
    ulong    vers;          //timestamp
} t_jit_bin_chunk_format_version;
typedef struct _jit_bin_chunk_matrix
{
    ulong    ckid;          //MTRX'
    long     cksize;        //varies(should be equal to
                           //24+(4*dimcount)+(typesize*planecount*totalpoints))
    long     offset;         //data offset(should be equal to 24+(4*dimcount))
    ulong    type;           //'CHAR','LONG','FL32','FL64'
    long     planecount;
    long     dimcount;
    long     dim[1];
} t_jit_bin_chunk_matrix;

```



## Chapter 30

# Jitter Networking Specification

This appendix describes the format of the data sent by a jit.net.send object.

The object attempts to form a TCP connection with a host at the IP and port specified by the object's attributes. Any program wishing to receive data will therefore have to set itself up as a host and listen for incoming TCP connections.

Once a connection is formed, data is sent as a stream of chunks. The first thing received will be a chunk header. It consists of a 32-bit chunk ID and a 32-bit int representing the size of the next chunk to come. The chunk ID can be one of the following 4-char symbols, depending on what kind of packet it is:

```
#define JIT_MATRIX_PACKET_ID 'JMTX'  
#define JIT_MATRIX_LATENCY_PACKET_ID 'JMPL'  
#define JIT_MESSAGE_PACKET_ID 'JMMP'
```

This chunk header could be represented in C by the following struct:

```
typedef struct _jit_net_packet_header  
{  
    t_int32 id;  
    t_int32 size; //size of packet to come  
} t_jit_net_packet_header;
```

If the chunk is a matrix packet, the next data received will be a header of 288 bytes with the following contents:

id	'JMTX'
Size	288 (32-bit int, size of this header)
PlaneCount	32-bit int
Type	32-bit int, 0 for char, 1 for long, 2 for float32, 3 for float64
DimCount	32-bit int
Dim	Array of 32 32-bit ints
DimStride	Array of 32 32-bit ints
DataSize	32-bit int, size of the data buffer to come
Time	64-bit double precision float

This chunk could be represented with the following C struct:

```
typedef struct _jit_net_packet_matrix  
{  
    t_int32 id;  
    t_int32 size;  
    t_int32 planeCount;  
    t_int32 type; //0=char,1=long,2=float32,3=float64  
    t_int32 dimCount;
```

```
t_int32 dim[JIT_MATRIX_MAX_DIMCOUNT];
t_int32 dimstride[JIT_MATRIX_MAX_DIMCOUNT];
t_int32 datasize;
double time;
} t_jit_net_packet_matrix;
```

Following this header the next data received will be the matrix data, the size of which was passed in the above header. When using the data, please note the dimstrides transmitted in the header.

The time field in the above header will be set to the time of transmission from the sending computer. jit.net.send expects the server to respond by sending back timing data of its own – it uses this data to estimate the transmission latency. The exact data in the latency chunk that jit.net.send expects to receive is the following:

<b>id</b>	'JMLP'
<b>client_time_original</b>	64-bit double, the time value received in the matrix header packet
<b>server_time_before_data</b>	64-bit double, the time on the server when the packet header is received
<b>server_time_after_data</b>	64-bit double, the time on the server after the packet has been processed and is in use

This chunk can be represented by the following C struct:

```
typedef struct _jit_net_packet_latency
{
    t_int32 id;
    double client_time_original;
    double server_time_before_data;
    double server_time_after_data;
} t_jit_net_packet_latency;
```

The difference between the server time before and server time after processing the data represents the time it takes the server to mobilize the data after it has been received. jit.net.send will send and expects to receive time in milliseconds. When this timing information is received by the transmitting computer, it notes its current time, calculates the round trip time and then estimates the latency as half the round trip time plus half of the server processing time. This estimate is accurate if the time of flight from A to B is the same as the time of flight from B to A, but network topology can be very complicated, and often the route from A to B is not the reverse of the route from B to A. In simple situations, such as a direct connection between two computers or a small LAN, the estimate should be reasonably accurate.

Finally, the last type of packet that can be sent is the message packet. The size of the message packet is sent in the initial header packet. Standard [A\\_GIMME](#) messages (**t\_symbol** \*s, long ac, **t\_atom** \*av) are serialized starting with a 32-bit integer that contains the size of the serialized message in bytes. Following that another 32-bit integer gives the argument count for the atoms. Following that comes the message atoms themselves, starting with the leading symbol if it exists. Each atom is represented in memory first with a char that indicates what type of atom it is: 's' for symbol, 'l' for long, and 'f' for float. For long and float atoms, the next 4 bytes contain the value of the atom; for symbol atoms a null terminated character string follows.

All data is represented with little endian byte ordering.

Below is a C function that will deserialize a message passed in as a data pointer.

```
void gimme_deserialize(char *data, t_symbol **s, long *ac, t_atom **av)
{
    char *curr = data;
    float *currf;
    long *currli;
    long datasize = BE_I32(*((long *)curr));
    curr += sizeof(long);
    *ac = BE_I32(*((long *)curr));
    curr += sizeof(long);
    *av = (t_atom *)sysmem_newptr(sizeof(t_atom)*(ac));
    if (*curr == ATOM_SERIALIZATION_SYMBOL_CODE)
    {
        curr++;
        *s = gensym(curr);
```

```

    while (*(++curr) != '\0') ;
    curr++;
}
else
    *s = 0L;
for (i=0;i<*ac;i++)
    switch (*curr++)
    {
        case ATOM_SERIALIZATION_SYMBOL_CODE:
            (*av)[i].a_type = A_SYM;
            (*av)[i].a_w.w_sym = gensym(curr);
            while (*(++curr) != '\0') ;
            curr++;
            break;
        case ATOM_SERIALIZATION_FLOAT_CODE:
            (*av)[i].a_type = A_FLOAT;
            (*av)[i].a_w.w_float = BE_F32(*((float *)curr));
            curr += sizeof(float);
            break;
        case ATOM_SERIALIZATION_LONG_CODE:
            (*av)[i].a_type = A_LONG;
            (*av)[i].a_w.w_long = BE_I32(*((long *)curr));
            curr += sizeof(long);
            break;
    }
}

```



## Chapter 31

# Appendix: Messages sent to Objects

When writing objects for Max, you typically think of creating methods which are called when a message is sent to your object through the object's inlet.

However, your object may receive messages directly from Max rather than using the inlet.

One common example is the "assist" message, which is sent to your object when a user's mouse cursor hovers over one of your object's inlets or outlets. If your object binds a method to the "assist" message then you will be able to customize the message that is shown.

This appendix serves as a quick reference for messages that are commonly sent to objects by Max, should they be implemented by the given object. Where possible, the prototypes given are actual prototypes from example objects in the SDK rather than abstractions to assist in finding the context for these calls.

### 31.1 Messages for All Objects

acceptsdrag_locked	long pictmeter_acceptsdrag_unlocked(t_↳ pictmeter *x, t_object *drag, t_object *view);	
acceptsdrag_unlocked	long pictmeter_acceptsdrag_unlocked(t_↳ pictmeter *x, t_object *drag, t_object *view);	
assist	void pictmeter_assist(t_pictmeter *x, void *b, long m, long a, char *s);	
dumpout		bind this message to <a href="#">object_obex_dumpout()</a> rather than defining your own method.
inletinfo	void my_obj(t_object *x, void *b, long a, char *t)	you may bind to stdinletinfo() or define your own inletinfo method.  The 'b' parameter can be ignored, the 'a' parameter is the inlet number, and 1 or 0 should set the value of '*t' upon return.

notify	t_max_err dbviewer_notify(t_dbviewer *x, t_symbol *s, t_symbol *msg, void *sender, void *data);	
stdargs	t_max_err my_obj(t_dictionary *d, t_symbol *s, long argc, t_atom *argv)	when loading an old (Max 3 or 4) patcher, this will be called prior to your new method. You can then fill in the dictionary with key/value pairs from your previous args.
quickref		obsolete, this is provided automatically now

## 31.2 Messages for Non-UI Objects

dblclick	void scripto_dblclick(t_scripto *x);	
----------	--------------------------------------	--

## 31.3 Messages for User Interface Objects

getdrawparams	void uisimp_getdrawparams(t_uisimp *x, t_object *patcherview, t_jboxdrawparams *params);	
mousedown	void scripto_ui_mousedown(t_scripto_ui *x, t_object *patcherview, t_pt pt, long modifiers);	
mouseup	void uisimp_mouseup(t_uisimp *x, t_object *patcherview, t_pt pt, long modifiers);	
mousedown	void scripto_ui_mousedown(t_scripto_ui *x, t_object *patcherview, t_pt pt, long modifiers);	
mouseenter	void uisimp_mouseenter(t_uisimp *x, t_object *patcherview, t_pt pt, long modifiers);	
mouseleave	void uisimp_mouseleave(t_uisimp *x, t_object *patcherview, t_pt pt, long modifiers);	
mousemove	void uisimp_mousemove(t_uisimp *x, t_object *patcherview, t_pt pt, long modifiers);	
paint	void pictmeter_paint(t_pictmeter *x, t_object *patcherview);	

## 31.4 Message for Audio Objects

dsp	void plus_dsp(t_plus *x, t_signal **sp, short *count);	
dspstate	plus_dspstate(t_plus *x, long n);	

## 31.5 Messages for Objects Containing Text Fields

key	long uittextfield_key(t_uittextfield *x, t_object *patcherview, long keycode, long modifiers, long textcharacter);	
keyfilter	long uittextfield_keyfilter(t_uittextfield *x, t_object *patcherview, long *keycode, long *modifiers, long *textcharacter);	
enter	void uittextfield_enter(t_uittextfield *x);	
select	void uittextfield_select(t_uittextfield *x);	

## 31.6 Messages for Objects with Text Editor Windows

edclose	void simpletext_edclose(t_simpletext *x, char **text, long size);	
---------	---	--

## 31.7 Messages for Dataview Client Objects

getcelltext	void dbviewer_getcelltext(t_dbviewer *x, t_symbol *colname, long index, char *text, long maxlen);	
newpatcherview	void dbviewer_newpatcherview(t_dbviewer *x, t_object *patcherview);	
freepatcherview	void dbviewer_freepatcherview(t_dbviewer *x, t_object *patcherview);	



## Chapter 32

# Appendix: Providing Icons for UI Objects

If you are writing user interface objects for Max, it is recommended that you provide an icon for your object.

Providing an icon will allow users to create an instance of your class from the object palette, and improve the user's experience in other interactions with Max including the Object Defaults inspector.

### 32.1 Object SVG Icon

To see the icons provided by Cycling '74 for objects included in Max, look in the **Cycling '74/object-icons** folder installed by Max. You will find a variety of SVG (scalable vector graphics) files for the objects. The files are named with the same name of the class (as it is defined in your `ext_main()` function) with which they are associated.

*SVG files can be edited in a variety of software applications such as InkScape or Adobe Illustrator. You can also export SVG files from OmniGraffle on the Mac, which is how the Max's object icons were created.*

It is recommended that you distribute your object as a Max Package (see [Appendix: SDK changes for Max 7](#)). Within this package you should place your svg in the 'interfaces' folder.

### 32.2 Quick Lookup Definition

Adding the svg file will make the icon available to Max for use in some ways. To make your icon appear in Max's Object Explorer, however, you must create a quick-lookup (or qlookup) entry for your object. If you look in the **Cycling '74/interfaces** folder, you should notice some files with names like "obj-qlookup.json" and "doc-qlookup.json". For your object, you should create a similar qlookup file.

For the following example we will assume you have created an object called 'littleuifoo'. For this object we will create a qlookup called 'littleuifoo-obj-qlookup.json'. The contents of this file will look like this:

```
{  
  "littleuifoo": {  
    "digest": "Little UI Object that does Foo",  
    "module": "max",  
    "category": [  
      "U/I"  
    ],  
    "palette": {  
      "category": [  
        "Interface"  
      ],  
      "action": "littleuifoo",  
      "pic": "littleuifoo.svg"  
    }  
  }  
}
```

## Chapter 33

# Appendix: Additional Resources

While it is out of the scope of this document to cover many topics related to Jitter development, we suggest the following resources to better inform your development.

The C Programming Language:

- "The C Programming Language", Kernighan and Ritchie (Prentice Hall, 1988). ISBN: 0131103709
- "A Book on C", Kelly and Pohl (Addison Wesley, 1997). ISBN: 0201183994
- [Wikipedia's C programming language resources](#)

Object Oriented Programming:

- [Wikipedia's Object Oriented Programming resources](#)
- [Sun's Object Oriented Programming Concepts Tutorial](#)
- [Object Oriented Programming in C](#)

Digital Image Processing:

- "Handbook of Image and Video Processing", A. Bovik et al. (Academic Press, 2000). ISBN: 0121197921
- "Digital Image Processing", W. K. Pratt (John Wiley and Sons, 2001). ISBN: 0471857661
- "Principles of Digital Image Synthesis", A. S. Glassner (Morgan Kaufmann, 1995). ISBN: 1558602763
- [Wikipedia's digital image processing resources](#)

Open GL:

- [The official OpenGL web portal](#)

Apple and QuickTime:

- [Apple's developer resources](#)

Microsoft:

- [Micrcosoft's developer resources](#)



## Chapter 34

# Appendix: Updating Externals for Max 6

## 34.1 Updating MSP Externals for 64-bit Audio Processing

### 34.1.1 Background

In Max 5 and prior versions, the signal chain for processing audio was compiled by sending all objects in the patcher a "dsp" message. Objects responding to this message then executed their dsp method, typically adding one of the object's perform methods to the signal chain.

In Max 6, the signal chain is compiled by first sending objects a "dsp64" message. When your object responds to this message, you can add your 64-bit audio perform methods. If an object supports the old "dsp" message but not the "dsp64" message, it then wraps the older 32-bit perform routine with conversion on the inputs and outputs.

This means that the 64-bit engine will work just fine with the older 32-bit objects. However, the conversion comes with some computational expense. For the best performance your objects should support the 64-bit dsp chain natively by implementing the "dsp64" message as explained below.

### 34.1.2 API

As noted, instead of the "dsp" method used by objects for Max 5 and earlier, Max 6 objects implement a "dsp64" method. This has the same purpose as the original dsp method. One notable difference is that the signals are not passed to the dsp64 method. This is to allow for the signal that is used to change dynamically at runtime. However, the relevant info (sample rate, number of signals connected, etc) is passed in.

The main purpose of the dsp64 method is to call back into the audio lib to put perform methods on the dsp chain. This is done by sending the 'dsp\_add64' message to the dspchain object using [object\\_method\(\)](#).

The perform routine is now of type t\_perfroutine64, defined in z\_dsp.h, and now has a fixed function signature. It does take a user-defined parameter that is passed back from the call to 'dsp\_add64'.

### 34.1.3 Example Code

The simplemsp~ examples in the 'audio' folder of the SDK have been updated for 64-bit audio processing in Max 6. Several projects, including the simplemsp~ example, demonstrate how to support both 64-bit audio processing in Max 6 and 32-bit audio processing for compatibility with Max 5.

## 34.2 Updating Max Externals for Cocoa

On the Macintosh platform, Max 6 made the transition from using the Carbon API to using the Cocoa API for interacting with the Mac OS. In most cases the transition for third-party developers should be seamless. If you are operating directly using native Carbon calls then your code will need to be updated to Cocoa using Objective-C.

The most common scenario is where you ask a patcherview for the native window handle with a call such as:

```
WindowRef viewWindow;
object_method(patcherview, gensym("nativewindow"), (void**)&viewWindow);
```

In Max 6 this will not work because the returned 'viewWindow' is not the Carbon WindowRef but is instead a Cocoa NSWindow\*. You may update your code to use Cocoa instead of Carbon, or you may wish to transition more slowly by continuing to use a WindowRef. Here is an example to assist in obtaining a WindowRef:

```
NSView      *cocoa_view = NULL;
NSWindow    *cocoa_window = NULL;
WindowRef   carbon_window;
object_method(patcherview, gensym("nativewindow"), (void**)&cocoa_view);
if (cocoa_view) {
    cocoa_window = [cocoa_view window];
    if (cocoa_window) {
        carbon_window = [cocoa_window windowRef];
    }
}
// now you can use your carbon_window as before
```

# Chapter 35

## Appendix: Updating Externals for Max 6.1 (x64 architecture)

### 35.1 Background

The Max 6.0.x application binary, and the external objects and libraries it uses, are compiled for the i386 processor architecture. This architecture uses 32-bit memory addressing, meaning that the size of a pointer is 32 bits (or 4 bytes).

Max 6.1 introduces support for the x86\_64 (or x64) architecture which uses 64-bit (8 bytes) memory addressing. Among the benefits are the ability to use more than 2 GB of memory in Max. Additionally, the size of the `t_atom` is 8-bytes on x64, meaning that double-precision floating pointer numbers can be represented.

For backwards compatibility, Max 6.1 also continues to be distributed as a 32-bit application binary. On the Windows platform the 32-bit and 64-bit applications are distributed separately, as are the external objects you create for them. On the Mac platform a Universal Binary (or "FAT" binary) is distributed containing both the 32-bit and 64-bit versions in the same dynamically-loaded library.

All externals on the Mac remain bundles using the ".mxo" filename extension.

32-bit externals on Windows remain DLLs using the ".mxe" filename extension.

64-bit externals on Windows are still DLLs but use a new ".mxe64" filename extension.

#### 35.1.1 New Types

In addition to the change of size in a pointer, there are some additional changes for 64-bit. For example, a "long" integer for 32-bit targets is 4 bytes on both the Mac and Windows. However, a "long" integer for 64-bit targets is 4 bytes on Windows but 8 bytes (the size of a pointer) on the Mac!

To facilitate cross platform code that is independent of these platform differences, the Max 6.1 API defines some new types used throughout the SDK.

Types of fixed size:

`t_int8`  
`t_uint8`

```
t_int16
t_uint16
t_int32
t_uint32
t_int64
t_uint64
```

#### Types of architecture-dependent size:

```
t_ptr_int : an int that is the same size as a pointer
t_ptr_uint : an unsigned int that is the same size as a pointer
t_atom_long : the type that is an A_LONG in an atom (32-bits on i386; 64-bits on x64)
t_atom_float : the type that is an A_FLOAT in an atom (float on i386; double on x64)
```

#### Types for specific contexts:

```
t_filepath : i.e. path/vol in file APIs identifying a folder
t_fourcc : use for type codes in locatefile_extended(), file dialogs, etc. to represent a four char code
t_getbytes_size : if you are using getbytes() and freebytes(), use this type to represent the size of the memory
```

## 35.2 Xcode and Visual Studio Projects

For new objects, you can base projects on those in the new SDK. To update existing projects you will need to make a few changes to your project settings.

### 35.2.1 Mac / Xcode

Max 6.1 on the Mac no longer uses the intermediary MaxAPI.framework for linking. Instead, the linking is handled at runtime and the symbols are checked using special flags to the linker. To update an existing project:

1. remove references to the MaxAPI.framework
2. update the .xcconfig file on which the project is based with the .xcconfig file in the new Max SDK
3. in your target's build settings find the "Other Linker Flags" and set it to "\$(C74\_SYM\_LINKER\_FLAGS)"
4. in your target's build settings find the "Architectures" and set it to "i386 x86\_64"

### 35.2.2 Windows / Visual Studio

In order to build for x64 with Visual Studio 2008, you must have the "Pro" version. The free "Express" version will not work. The "Express" versions of Visual Studio 2010 and 2012 do work (2012 is recommended).

Due to bugs in Visual Studio 2008, it is really difficult to update an existing project. Instead, it is recommended to simply create a new Visual Studio project based on an existing example. For Visual Studio 2008 use the "vcproj" files. For Visual Studio 2010 and 2012 use the "vcxproj" files.

1. choose a relevant starting point, e.g., the "dummy" example project
2. copy it to the folder your old project was in, rename it
3. open it in a text editor such as "sublime text 2"
4. do a find/replace for all instances of the text "dummy" changing it to your object's name
5. open the Visual Studio project and build you can choose either "Win32" or "x64" from the platform drop-down menu in the IDE

## 35.3 Changes to Code

### 35.3.1 Atoms

Any assumptions in your code about the size of a `t_atom` or the size of its members should be reviewed. When setting or getting values to and from atoms you should use the types `t_atom_long` and `t_atom_float` as appropriate.

### 35.3.2 Return Values

All methods which return a value must return a pointer-sized value, e.g., `t_ptr_int`, `t_ptr_uint`, `t_max_err`, etc.

### 35.3.3 File Code

File access in Max involves several areas subject to either required or suggested update.

A path in Max has traditionally been represented with a short int; it is recommended to now use the new `t_filepath` type.

File types in Max are represented using four char codes. Traditionally these have been defined using variables of type "long", which is now problematic. This is a 4-byte type but the long on the Mac for x64 is 8-bytes. These must be updated to use the new `t_fourcc` type.

### 35.3.4 Miscellaneous Gotchas

Look for any ld or lu strings in sprintf() or related formatting functions.

## 35.4 Common Scenarios

### 35.4.1 "Long" integers

One of biggest areas we've had to address is the use of the long datatype. The reason for this is that under 64bit windows a long integer is 32 bits and under 64bit OS X (and Unix), a long is a 64 bit integer.

To assist in this process, we have the new data types documented above. We'll distinguish these from what we are calling a "platform long".

This platform long discrepancy can lead to all sorts of problems which are outlined with a brief statement of the problem scenario, and our recommended fix with types:

**Problem: long integers as A\_LONG/A\_DEFLONG method arguments (this includes your object constructors)**

Solution: type your A\_LONG/A\_DEFLONG methods' function signatures to use the `t_atom_long` in place of long

**Problem: long integers as A\_CANT method arguments called only through `object_method()`**

Solution: either redefine your A\_CANT method's arguments to t\_atom\_long, or define your type as A\_DIRECT, and make use of the [object\\_method\\_direct\(\)](#) macro, passing in a function prototype to the macro (also see under floating point how this is required for anything which previously was A\_CANT with floating point values). Technically many of these will still work properly due to the nature of how integers are passed on the stack under x64, without any change, it is still best practice.

**Problem: long integers being used to store pointers as integer values either for pointer arithmetic, attributes, or other situations.**

Solution: use t\_atom\_long or even better t\_ptr\_uint (for pointer sized unsigned integer) or the actual pointer type.

**Problem: long integers as four character codes for filetype (t\_fourcc) which applies to locatefile\_extended and path functions and friends**

Solution: Use t\_foucc inplace of long, for anywhere you are using filetype codes.

**Problem: long integers as return values for functions called via [object\\_method\(\)](#)**

Solution: These should always return a t\_atom\_long or other pointer sized integer

**Problem: long integers passed as pointers into functions like [dictionary\\_getlong\(\)](#) which are now prototyped to take a t\_atom\_long \***

Solution: Use a t\_atom\_long value, and pass a pointer to it. A cast from a platform long \*to a t\_atom\_long \* is not safe.

**Problem: long integers for performing bitwise manipulation of 32bit floating point values including byteswapping**

Solution: Use t\_int32/t\_uint32

There are many cases where it is safe to use long integers, and we have continued to use them in our code. Below are the scenarios where they are okay and in several cases required. This might provide some confusion at some points, but hopefully it makes the porting process a little bit easier, allowing more code to remain unchanged.

- Attributes defined as \_sym\_long should remain a platform long. If you need to have a t\_atom\_long attribute, you will need to use the new atom\_long attribute type. This is probably the most confusing aspect of porting to 64bit and a very real ambiguity of the word "long". Unfortunately, having to balance the difficulties of porting with the clarity of API, this is something we felt necessary to do.
- Attribute getters/setters should still use the long type for ac. this is especially important for getters which are passed a pointer to a platform long in the ac value.
- A\_GIMME methods may still use the long type for ac without issues

### 35.4.2 Floating point values

For A\_FLOAT/A\_DEFFLOAT function signatures, you should always use double as is currently recommended in the Max SDK. You should not use the new t\_atom\_float datatype. (this includes your object constructors)

For A\_CANT functions with floating point arguments that currently use [object\\_method\(\)](#). You will need to use [object\\_method\\_direct\(\)](#) or pass in pointers to the floating point values (which is safe as it is a pointer sized integer). It is no longer possible to pass floats through [object\\_method\(\)](#) or the many functions like it ([linklist\\_methodall\(\)](#), [hashtab\\_methodall\(\)](#), etc.)

Attributes are already defined in terms of their bitsize float32 or float 64. If you wish for your attribute to make use of the new atom support for double precision. You will want to change your struct definition, as well as your attribute constructor to be a double (\_sym\_float64). There isn't currently a t\_atom\_float attribute type like we've added for long.

### 35.4.3 Deprecated Apple types

like Byte/Boolean/Point/etc

Use the t\_byte/t\_bool/t\_point/etc types instead.

## 35.5 Additional Miscellaneous Changes

The old 32-bit 'dsp' (and perform) methods are no longer supported as of Max 6.1. They must be updated as per [Updating MSP Externals for 64-bit Audio Processing](#).

## 35.6 Additional Resources

- <http://www.viva64.com/en/a/0004/>
- <https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/Cocoa64BitGuide/64BitChangesCocoa/64BitChangesCocoa.html>
- <https://developer.apple.com/library/mac/#documentation/Carbon/Conceptual/Carbon64BitGuide/OtherAPIChanges/OtherAPIChanges.html>
- <http://msdn.microsoft.com/en-us/magazine/cc300794.aspx>



## Chapter 36

# Appendix: SDK changes for Max 7

### 36.1 Using `ext_main` for Entry Point

Prior to Max 7 the entry point for externals was the `main()` function exported from the dynamic library you create. Beginning with Max 7 the entry point for externals is called `ext_main()`. This addresses compatibility problems with various newer compilers and frees us from the constraints enforced for `main()` as the standard entry point for programs.

Objects that do not define `ext_main()` will still be loaded using the older `main()`. Support for `ext_main()` is also present in Max 6.1.9.

### 36.2 Support for Max 7 UI Object Styles

Max 7 introduces the concept of styles which determine the appearance of UI objects. For attributes of your UI object to map to colors or attributes of a style you need to add the required attribute properties in your class definition.

See the section on [Styles](#) for more information.



## Chapter 37

# Appendix: SDK changes for Max 8

### 37.1 Support for MC

Max 8 introduces the concept of MC for multi-voice/multi-channel signal processing. See the chapter on [MC](#) for more information.



# Chapter 38

## Module Documentation

### 38.1 Attributes

An attribute of an object is a setting or property that tells the object how to do its job.

#### Data Structures

- struct `t_attr`

*Common attr struct.*

#### Macros

- `#define CLASS_ATTR_OFFSET_DUMMY(c, attrname, flags, typesym)`  
*Create an attribute that does not store its data in the object struct.*
- `#define CLASS_ATTR_CHAR(c, attrname, flags, structname, structmember)`  
*Create a char attribute and add it to a Max class.*
- `#define CLASS_ATTR_LONG(c, attrname, flags, structname, structmember)`  
*Create a long integer attribute and add it to a Max class.*
- `#define CLASS_ATTR_ATOM_LONG(c, attrname, flags, structname, structmember)`  
*Create a t\_atom\_long integer attribute and add it to a Max class.*
- `#define CLASS_ATTR_INT32(c, attrname, flags, structname, structmember)`  
*Create a t\_int32 integer attribute and add it to a Max class.*
- `#define CLASS_ATTR_FLOAT(c, attrname, flags, structname, structmember)`  
*Create a 32-bit float attribute and add it to a Max class.*
- `#define CLASS_ATTR_DOUBLE(c, attrname, flags, structname, structmember)`  
*Create a 64-bit float attribute and add it to a Max class.*
- `#define CLASS_ATTR_SYMBOL(c, attrname, flags, structname, structmember)`  
*Create a t\_symbol\* attribute and add it to a Max class.*
- `#define CLASS_ATTR_ATOM(c, attrname, flags, structname, structmember)`  
*Create a t\_atom attribute and add it to a Max class.*

- #define **CLASS\_ATTR\_OBJ**(c, attrname, flags, structname, structmember)
 

*Create a `t_object`\* attribute and add it to a Max class.*
- #define **CLASS\_ATTR\_CHAR\_ARRAY**(c, attrname, flags, structname, structmember, size)
 

*Create an array-of-chars attribute of fixed length, and add it to a Max class.*
- #define **CLASS\_ATTR\_LONG\_ARRAY**(c, attrname, flags, structname, structmember, size)
 

*Create an array-of-long-integers attribute of fixed length, and add it to a Max class.*
- #define **CLASS\_ATTR\_ATOM\_LONG\_ARRAY**(c, attrname, flags, structname, structmember, size)
 

*Create an array-of-`t_atom_long`-integers attribute of fixed length, and add it to a Max class.*
- #define **CLASS\_ATTR\_FLOAT\_ARRAY**(c, attrname, flags, structname, structmember, size)
 

*Create an array-of-32bit-floats attribute of fixed length, and add it to a Max class.*
- #define **CLASS\_ATTR\_DOUBLE\_ARRAY**(c, attrname, flags, structname, structmember, size)
 

*Create an array-of-64bit-floats attribute of fixed length, and add it to a Max class.*
- #define **CLASS\_ATTR\_SYM\_ARRAY**(c, attrname, flags, structname, structmember, size)
 

*Create an array-of-symbols attribute of fixed length, and add it to a Max class.*
- #define **CLASS\_ATTR\_ATOM\_ARRAY**(c, attrname, flags, structname, structmember, size)
 

*Create an array-of-atoms attribute of fixed length, and add it to a Max class.*
- #define **CLASS\_ATTR\_OBJ\_ARRAY**(c, attrname, flags, structname, structmember, size)
 

*Create an array-of-objects attribute of fixed length, and add it to a Max class.*
- #define **CLASS\_ATTR\_CHAR\_VARSIZE**(c, attrname, flags, structname, structmember, sizemember, maxsize)
 

*Create an array-of-chars attribute of variable length, and add it to a Max class.*
- #define **CLASS\_ATTR\_LONG\_VARSIZE**(c, attrname, flags, structname, structmember, sizemember, maxsize)
 

*Create an array-of-long-integers attribute of variable length, and add it to a Max class.*
- #define **CLASS\_ATTR\_FLOAT\_VARSIZE**(c, attrname, flags, structname, structmember, sizemember, maxsize)
 

*Create an array-of-32bit-floats attribute of variable length, and add it to a Max class.*
- #define **CLASS\_ATTR\_DOUBLE\_VARSIZE**(c, attrname, flags, structname, structmember, sizemember, maxsize)
 

*Create an array-of-64bit-floats attribute of variable length, and add it to a Max class.*
- #define **CLASS\_ATTR\_SYM\_VARSIZE**(c, attrname, flags, structname, structmember, sizemember, maxsize)
 

*Create an array-of-symbols attribute of variable length, and add it to a Max class.*
- #define **CLASS\_ATTR\_ATOM\_VARSIZE**(c, attrname, flags, structname, structmember, sizemember, maxsize)
 

*Create an array-of-atoms attribute of variable length, and add it to a Max class.*
- #define **CLASS\_ATTR\_OBJ\_VARSIZE**(c, attrname, flags, structname, structmember, sizemember, maxsize)
 

*Create an array-of-objects attribute of variable length, and add it to a Max class.*
- #define **STRUCT\_ATTR\_CHAR**(c, flags, structname, structmember)
 

*Create a char attribute and add it to a Max class.*
- #define **STRUCT\_ATTR\_LONG**(c, flags, structname, structmember)
 

*Create a long integer attribute and add it to a Max class.*
- #define **STRUCT\_ATTR\_ATOM\_LONG**(c, flags, structname, structmember)
 

*Create a `t_atom_long` integer attribute and add it to a Max class.*
- #define **STRUCT\_ATTR\_FLOAT**(c, flags, structname, structmember)
 

*Create a 32bit float attribute and add it to a Max class.*
- #define **STRUCT\_ATTR\_DOUBLE**(c, flags, structname, structmember)
 

*Create a 64bit float attribute and add it to a Max class.*
- #define **STRUCT\_ATTR\_SYM**(c, flags, structname, structmember)
 

*Create a `t_symbol`\* attribute and add it to a Max class.*
- #define **STRUCT\_ATTR\_ATOM**(c, flags, structname, structmember)
 

*Create a `t_atom` attribute and add it to a Max class.*
- #define **STRUCT\_ATTR\_OBJ**(c, flags, structname, structmember)

- Create a `t_object*` attribute and add it to a Max class.  
`#define STRUCT_ATTR_CHAR_ARRAY(c, flags, structname, structmember, size)`  
*Create an array-of-chars attribute of fixed length, and add it to a Max class.*
- `#define STRUCT_ATTR_LONG_ARRAY(c, flags, structname, structmember, size)`  
*Create an array-of-long-integers attribute of fixed length, and add it to a Max class.*
- `#define STRUCT_ATTR_FLOAT_ARRAY(c, flags, structname, structmember, size)`  
*Create an array-of-32bit-floats attribute of fixed length, and add it to a Max class.*
- `#define STRUCT_ATTR_DOUBLE_ARRAY(c, flags, structname, structmember, size)`  
*Create an array-of-64bit-floats attribute of fixed length, and add it to a Max class.*
- `#define STRUCT_ATTR_SYM_ARRAY(c, flags, structname, structmember, size)`  
*Create an array-of-symbols attribute of fixed length, and add it to a Max class.*
- `#define STRUCT_ATTR_ATOM_ARRAY(c, flags, structname, structmember, size)`  
*Create an array-of-atoms attribute of fixed length, and add it to a Max class.*
- `#define STRUCT_ATTR_OBJ_ARRAY(c, flags, structname, structmember, size)`  
*Create an array-of-objects attribute of fixed length, and add it to a Max class.*
- `#define STRUCT_ATTR_CHAR_VARSIZE(c, flags, structname, structmember, sizemember, maxsize)`  
*Create an array-of-chars attribute of variable length, and add it to a Max class.*
- `#define STRUCT_ATTR_LONG_VARSIZE(c, flags, structname, structmember, sizemember, maxsize)`  
*Create an array-of-long-integers attribute of variable length, and add it to a Max class.*
- `#define STRUCT_ATTR_FLOAT_VARSIZE(c, flags, structname, structmember, sizemember, maxsize)`  
*Create an array-of-32bit-floats attribute of variable length, and add it to a Max class.*
- `#define STRUCT_ATTR_DOUBLE_VARSIZE(c, flags, structname, structmember, sizemember, maxsize)`  
*Create an array-of-64bit-floats attribute of variable length, and add it to a Max class.*
- `#define STRUCT_ATTR_SYM_VARSIZE(c, flags, structname, structmember, sizemember, maxsize)`  
*Create an array-of-symbols attribute of variable length, and add it to a Max class.*
- `#define STRUCT_ATTR_ATOM_VARSIZE(c, flags, structname, structmember, sizemember, maxsize)`  
*Create an array-of-atoms attribute of variable length, and add it to a Max class.*
- `#define STRUCT_ATTR_OBJ_VARSIZE(c, flags, structname, structmember, sizemember, maxsize)`  
*Create an array-of-objects attribute of variable length, and add it to a Max class.*
- `#define OBJ_ATTR_CHAR(x, attrname, flags, val)`  
*Create an instance-local char attribute and add it to a Max class.*
- `#define OBJ_ATTR_LONG(x, attrname, flags, val)`  
*Create an instance-local long integer attribute and add it to a Max class.*
- `#define OBJ_ATTR_FLOAT(x, attrname, flags, val)`  
*Create an instance-local 32bit float attribute and add it to a Max class.*
- `#define OBJ_ATTR_DOUBLE(x, attrname, flags, val)`  
*Create an instance-local 64bit float attribute and add it to a Max class.*
- `#define OBJ_ATTR_SYM(x, attrname, flags, val)`  
*Create an instance-local `t_symbol*` attribute and add it to a Max class.*
- `#define OBJ_ATTR_ATOM(x, attrname, flags, val)`  
*Create an instance-local `t_atom` attribute and add it to a Max class.*
- `#define OBJ_ATTR_OBJ(x, attrname, flags, val)`  
*Create an instance-local `t_object*` attribute and add it to a Max class.*
- `#define OBJ_ATTR_CHAR_ARRAY(x, attrname, flags, count, vals)`  
*Create an instance-local array-of-chars attribute of fixed length, and add it to the object.*
- `#define OBJ_ATTR_LONG_ARRAY(x, attrname, flags, count, vals)`  
*Create an instance-local array-of-long-integers attribute of fixed length, and add it to the object.*

- `#define OBJ_ATTR_FLOAT_ARRAY(x, attrname, flags, count, vals)`  
*Create an instance-local array-of-32bit-floats attribute of fixed length, and add it to the object.*
- `#define OBJ_ATTR_DOUBLE_ARRAY(x, attrname, flags, count, vals)`  
*Create an instance-local array-of-64bit-floats attribute of fixed length, and add it to the object.*
- `#define OBJ_ATTR_SYM_ARRAY(x, attrname, flags, count, vals)`  
*Create an instance-local array-of-symbols attribute of fixed length, and add it to the object.*
- `#define OBJ_ATTR_ATOM_ARRAY`  
*Create an instance-local array-of-atoms attribute of fixed length, and add it to the object.*
- `#define OBJ_ATTR_OBJ_ARRAY(x, attrname, flags, count, vals)`  
*Create an instance-local array-of-objects attribute of fixed length, and add it to the object.*
- `#define CLASS_ATTR_ACCESSORS(c, attrname, getter, setter)`  
*Specify custom accessor methods for an attribute.*
- `#define CLASS_ATTR_ADD_FLAGS(c, attrname, flags)`  
*Add flags to an attribute.*
- `#define CLASS_ATTR_REMOVE_FLAGS(c, attrname, flags)`  
*Remove flags from an attribute.*
- `#define CLASS_ATTR_FILTER_MIN(c, attrname, minval)`  
*Add a filter to the attribute to limit the lower bound of a value.*
- `#define CLASS_ATTR_FILTER_MAX(c, attrname, maxval)`  
*Add a filter to the attribute to limit the upper bound of a value.*
- `#define CLASS_ATTR_FILTER_CLIP(c, attrname, minval, maxval)`  
*Add a filter to the attribute to limit both the lower and upper bounds of a value.*
- `#define CLASS_ATTR_ALIAS(c, attrname, aliasname)`  
*Create a new attribute that is an alias of an existing attribute.*
- `#define CLASS_ATTR_DEFAULT(c, attrname, flags, parsestr)`  
*Add a new attribute to the specified attribute to specify a default value.*
- `#define CLASS_ATTR_SAVE(c, attrname, flags)`  
*Add a new attribute to the specified attribute to indicate that the specified attribute should be saved with the patcher.*
- `#define CLASS_ATTR_SELFSAVE(c, attrname, flags)`  
*Add a new attribute to the specified attribute to indicate that it is saved by the object (so it does not appear in italics in the inspector).*
- `#define CLASS_ATTR_DEFAULT_SAVE(c, attrname, flags, parsestr)`  
*A convenience wrapper for both `CLASS_ATTR_DEFAULT` and `CLASS_ATTR_SAVE`.*
- `#define CLASS_ATTR_DEFAULTNAME(c, attrname, flags, parsestr)`  
*Add a new attribute to the specified attribute to specify a default value, based on Max's Object Defaults.*
- `#define CLASS_ATTR_DEFAULTNAME_SAVE(c, attrname, flags, parsestr)`  
*A convenience wrapper for both `CLASS_ATTR_DEFAULTNAME` and `CLASS_ATTR_SAVE`.*
- `#define CLASS_ATTR_MIN(c, attrname, flags, parsestr)`  
*Add a new attribute to the specified attribute to specify a lower range.*
- `#define CLASS_ATTR_MAX(c, attrname, flags, parsestr)`  
*Add a new attribute to the specified attribute to specify an upper range.*
- `#define CLASS_ATTR_PAINT(c, attrname, flags)`  
*Add a new attribute indicating that any changes to the specified attribute will trigger a call to the object's paint method.*
- `#define CLASS_ATTR_DEFAULT_PAINT(c, attrname, flags, parsestr)`  
*A convenience wrapper for both `CLASS_ATTR_DEFAULT` and `CLASS_ATTR_PAINT`.*
- `#define CLASS_ATTR_DEFAULT_SAVE_PAINT(c, attrname, flags, parsestr)`  
*A convenience wrapper for `CLASS_ATTR_DEFAULT`, `CLASS_ATTR_SAVE`, and `CLASS_ATTR_PAINT`.*

- `#define CLASS_ATTR_DEFAULTNAME_PAINT(c, attrname, flags, parsestr)`  
*A convenience wrapper for `CLASS_ATTR_DEFAULTNAME`, `CLASS_ATTR_SAVE`, and `CLASS_ATTR_PAINT`.*
- `#define CLASS_ATTR_DEFAULTNAME_SAVE_PAINT(c, attrname, flags, parsestr)`  
*A convenience wrapper for `CLASS_ATTR_DEFAULTNAME`, `CLASS_ATTR_SAVE`, and `CLASS_ATTR_PAINT`.*
- `#define CLASS_ATTR_STYLE(c, attrname, flags, parsestr)`  
*Add a new attribute to the specified attribute to specify an editor style for the Max inspector.*
- `#define CLASS_ATTR_LABEL(c, attrname, flags, labelstr)`  
*Add a new attribute to the specified attribute to specify an a human-friendly label for the Max inspector.*
- `#define CLASS_ATTR_ENUM(c, attrname, flags, parsestr)`  
*Add a new attribute to the specified attribute to specify a list of choices to display in a menu for the Max inspector.*
- `#define CLASS_ATTR_ENUMINDEX(c, attrname, flags, parsestr)`  
*Add a new attribute to the specified attribute to specify a list of choices to display in a menu for the Max inspector.*
- `#define CLASS_ATTR_CATEGORY(c, attrname, flags, parsestr)`  
*Add a new attribute to the specified attribute to specify a category to which the attribute is assigned in the Max inspector.*
- `#define CLASS_ATTR_STYLE_LABEL(c, attrname, flags, stylestr, labelstr)`  
*A convenience wrapper for `CLASS_ATTR_STYLE`, and `CLASS_ATTR_LABEL`.*
- `#define CLASS_ATTR_INVISIBLE(c, attrname, flags)`  
*Add a new attribute to the specified attribute to flag an attribute as invisible to the Max inspector.*
- `#define CLASS_ATTR_ORDER(c, attrname, flags, parsestr)`  
*Add a new attribute to the specified attribute to specify a default order in which to list attributes.*
- `#define CLASS_ATTR_BASIC(c, attrname, flags)`  
*Add a new attribute to the specified attribute to specify that it should appear in the inspector's Basic tab.*
- `#define CLASS_METHOD_ATTR_PARSE(c, methodname, attrname, type, flags, parsestring)`  
*Define and add attributes to class methods.*
- `#define CLASS_ATTR_LEGACYDEFAULT(c, legacyattrname, newattrname, flags, parsestr)`  
*Add a new attribute to the specified attribute to specify a legacy default value.*
- `#define CLASS_ATTR_OBSOLETE(c, attrname, flags)`  
*Add a new attribute to the specified attribute to flag it as obsolete.*
- `#define CLASS_ATTR_RENAMED(c, oldname, newname, flags)`  
*Add a new attribute to the specified attribute to flag it as renamed.*
- `#define CLASS_ATTR_INTRODUCED(c, attrname, flags, versionstr)`  
*Add a new attribute to the specified attribute to indicate in which version the attribute was introduced.*
- `#define CLASS_METHOD_OBSOLETE(c, methodname, flags)`  
*Add a new attribute to the specified method to flag it as obsolete.*
- `#define CLASS_METHOD_RENAMED(c, oldname, newname, flags)`  
*Add a new attribute to the specified method to flag a method as renamed.*
- `#define CLASS_METHOD_INTRODUCED(c, methodname, flags, versionstr)`  
*Add a new attribute to the specified method to indicate in which version the method was introduced.*
- `#define OBJ_ATTR_DEFAULT(x, attrname, flags, parsestr)`  
*An instance-attribute version of `CLASS_ATTR_DEFAULT`.*
- `#define OBJ_ATTR_SAVE(x, attrname, flags)`  
*An instance-attribute version of `CLASS_ATTR_SAVE`.*
- `#define OBJ_ATTR_DEFAULT_SAVE(x, attrname, flags, parsestr)`  
*An instance-attribute version of `CLASS_ATTR_DEFAULT_SAVE`.*
- `#define CLASS_STICKY_ATTR(c, name, flags, parsestr)`  
*Create an attribute, and add it to all following attribute declarations.*
- `#define CLASS_STICKY_ATTR_CLEAR(c, name)`

- `#define CLASS_STICKY_METHOD(c, name, flags, parsestr)`  
*Create an attribute, and add it to all following method declarations.*
- `#define CLASS_STICKY_METHOD_CLEAR(c, name)`  
*Close a CLASS\_STICKY\_METHOD block.*
- `#define CLASS_ATTR_RGBA(c, attrname, flags, structname, structmember)`  
*Create a color (`t_jrgba`) attribute and add it to a Max class.*

## Enumerations

- `enum e_max_attrflags { ATTR_FLAGS_NONE, ATTR_GET_OPAQUE, ATTR_SET_OPAQUE, ATTR_GET_OPAQUE_USER, ATTR_SET_OPAQUE_USER, ATTR_GET_DEFER, ATTR_GET_USURP, ATTR_GET_DEFER_LOW, ATTR_GET_USURP_LOW, ATTR_SET_DEFER, ATTR_SET_USURP, ATTR_SET_DEFER_LOW, ATTR_SET_USURP_LOW, ATTR_IS_JBOXATTR, ATTR_DIRTY }`  
*Attribute flags.*

## Functions

- `void * object_attr_get (void **x, t_symbol *attrname)`  
*Returns the pointer to an attribute, given its name.*
- `method object_attr_method (void **x, t_symbol *methodname, void **attr, long *get)`  
*Returns the method of an attribute's get or set function, as well as a pointer to the attribute itself, from a message name.*
- `long object_attr_usercanset (void **x, t_symbol *s)`  
*Determines if an object's attribute can be set from the Max interface (i.e.*
- `long object_attr_usercanget (void **x, t_symbol *s)`  
*Determines if the value of an object's attribute can be queried from the Max interface (i.e.*
- `void object_attr_getdump (void **x, t_symbol *s, long argc, t_atom *argv)`  
*Forces a specified object's attribute to send its value from the object's dumpout outlet in the Max interface.*
- `t_max_err object_attr_setvalueof (void **x, t_symbol *s, long argc, t_atom *argv)`  
*Sets the value of an object's attribute.*
- `t_max_err object_addattr (void **x, t_object *attr)`  
*Attaches an attribute directly to an object.*
- `t_max_err object_deleteattr (void **x, t_symbol *attrsym)`  
*Detach an attribute from an object that was previously attached with `object_addattr()`.*
- `t_max_err object_chuckattr (void **x, t_symbol *attrsym)`  
*Detach an attribute from an object that was previously attached with `object_addattr()`.*
- `long attr_args_offset (short ac, t_atom *av)`  
*Determines the point in an atom list where attribute arguments begin.*
- `void attr_args_process (void **x, short ac, t_atom *av)`  
*Takes an atom list and properly set any attributes described within.*
- `t_object * attribute_new (C74_CONST char *name, t_symbol *type, long flags, method mget, method mset)`  
*Create a new attribute.*
- `t_object * attr_offset_new (C74_CONST char *name, C74_CONST t_symbol *type, long flags, C74_CONST method mget, C74_CONST method mset, long offset)`  
*Create a new attribute.*

- `t_object * attr_offset_array_new (C74_CONST char *name, t_symbol *type, long size, long flags, method mget, method mset, long offsetcount, long offset)`

*Create a new attribute.*
- `t_atom_long object_attr_getlong (void *x, t_symbol *s)`

*Retrieves the value of an attribute, given its parent object and name.*
- `t_max_err object_attr_setlong (void *x, t_symbol *s, t_atom_long c)`

*Sets the value of an attribute, given its parent object and name.*
- `t_atom_float object_attr_getfloat (void *x, t_symbol *s)`

*Retrieves the value of an attribute, given its parent object and name.*
- `t_max_err object_attr_setfloat (void *x, t_symbol *s, t_atom_float c)`

*Sets the value of an attribute, given its parent object and name.*
- `t_symbol * object_attr_getsym (void *x, t_symbol *s)`

*Retrieves the value of an attribute, given its parent object and name.*
- `t_max_err object_attr_setsym (void *x, t_symbol *s, t_symbol *c)`

*Sets the value of an attribute, given its parent object and name.*
- `long object_attr_getlong_array (void *x, t_symbol *s, long max, t_atom_long *vals)`

*Retrieves the value of an attribute, given its parent object and name.*
- `t_max_err object_attr_setlong_array (void *x, t_symbol *s, long count, t_atom_long *vals)`

*Sets the value of an attribute, given its parent object and name.*
- `long object_attr_getchar_array (void *x, t_symbol *s, long max, t_uint8 *vals)`

*Retrieves the value of an attribute, given its parent object and name.*
- `t_max_err object_attr_setchar_array (void *x, t_symbol *s, long count, C74_CONST t_uint8 *vals)`

*Sets the value of an attribute, given its parent object and name.*
- `long object_attr_getfloat_array (void *x, t_symbol *s, long max, float *vals)`

*Retrieves the value of an attribute, given its parent object and name.*
- `t_max_err object_attr_setfloat_array (void *x, t_symbol *s, long count, float *vals)`

*Sets the value of an attribute, given its parent object and name.*
- `long object_attr_getdouble_array (void *x, t_symbol *s, long max, double *vals)`

*Retrieves the value of an attribute, given its parent object and name.*
- `t_max_err object_attr_setdouble_array (void *x, t_symbol *s, long count, double *vals)`

*Sets the value of an attribute, given its parent object and name.*
- `long object_attr_getsym_array (void *x, t_symbol *s, long max, t_symbol **vals)`

*Retrieves the value of an attribute, given its parent object and name.*
- `t_max_err object_attr_setsym_array (void *x, t_symbol *s, long count, t_symbol **vals)`

*Sets the value of an attribute, given its parent object and name.*
- `t_max_err attr_addfilterset_clip (void *x, double min, double max, long usemin, long usemax)`

*Attaches a clip filter to an attribute.*
- `t_max_err attr_addfilterset_clip_scale (void *x, double scale, double min, double max, long usemin, long usemax)`

*Attaches a clip/scale filter to an attribute.*
- `t_max_err attr_addfilterget_clip (void *x, double min, double max, long usemin, long usemax)`

*Attaches a clip filter to an attribute.*
- `t_max_err attr_addfilterget_clip_scale (void *x, double scale, double min, double max, long usemin, long usemax)`

*Attaches a clip/scale filter to an attribute.*
- `t_max_err attr_addfilter_clip (void *x, double min, double max, long usemin, long usemax)`

*Attaches a clip filter to an attribute.*
- `t_max_err attr_addfilter_clip_scale (void *x, double scale, double min, double max, long usemin, long usemax)`

*Attaches a clip/scale filter to an attribute.*

- `t_max_err attr_addfilterset_proc` (void \*x, method proc)  
*Attaches a custom filter method to an attribute.*
- `t_max_err attr_addfilterget_proc` (void \*x, method proc)  
*Attaches a custom filter method to an attribute.*
- void `attr_args_dictionary` (`t_dictionary` \*x, short ac, `t_atom` \*av)  
*Create a dictionary of attribute-name, attribute-value pairs from an array of atoms containing an attribute definition list.*
- void `attr_dictionary_process` (void \*x, `t_dictionary` \*d)  
*Set attributes for an object that are defined in a dictionary.*
- void `attr_dictionary_check` (void \*x, `t_dictionary` \*d)  
*Check that a dictionary only contains values for existing attributes of an object.*
- `t_max_err object_attr_setparse` (`t_object` \*x, `t_symbol` \*s, C74\_CONST char \*parsestr)  
*Set an attribute value with one or more atoms parsed from a C-string.*
- void \* `object_new_parse` (`t_symbol` \*name\_space, `t_symbol` \*classname, C74\_CONST char \*parsestr)  
*Create a new object with one or more atoms parsed from a C-string.*
- `t_max_err object_attr_getjrgba` (void \*ob, `t_symbol` \*s, `t_jrgba` \*c)  
*Retrieves the value of a color attribute, given its parent object and name.*
- `t_max_err object_attr_setjrgba` (void \*ob, `t_symbol` \*s, `t_jrgba` \*c)  
*Sets the value of a color attribute, given its parent object and name.*
- `t_max_err object_attr_get_rect` (`t_object` \*o, `t_symbol` \*name, `t_rect` \*rect)  
*Gets the value of a `t_rect` attribute, given its parent object and name.*
- `t_max_err object_attr_set_rect` (`t_object` \*o, `t_symbol` \*name, `t_rect` \*rect)  
*Sets the value of a `t_rect` attribute, given its parent object and name.*
- void `object_attr_set_xywh` (`t_object` \*o, `t_symbol` \*attr, double x, double y, double w, double h)  
*Sets the value of a `t_rect` attribute, given its parent object and name.*
- `t_max_err object_attr_getpt` (`t_object` \*o, `t_symbol` \*name, `t_pt` \*pt)  
*Gets the value of a `t_pt` attribute, given its parent object and name.*
- `t_max_err object_attr_setpt` (`t_object` \*o, `t_symbol` \*name, `t_pt` \*pt)  
*Sets the value of a `t_pt` attribute, given its parent object and name.*
- `t_max_err object_attr_getsize` (`t_object` \*o, `t_symbol` \*name, `t_size` \*size)  
*Gets the value of a `t_size` attribute, given its parent object and name.*
- `t_max_err object_attr_setsize` (`t_object` \*o, `t_symbol` \*name, `t_size` \*size)  
*Sets the value of a `t_size` attribute, given its parent object and name.*
- `t_max_err object_attr_getcolor` (`t_object` \*b, `t_symbol` \*attrname, `t_jrgba` \*prgba)  
*Gets the value of a `t_jrgba` attribute, given its parent object and name.*
- `t_max_err object_attr_setcolor` (`t_object` \*b, `t_symbol` \*attrname, `t_jrgba` \*prgba)  
*Sets the value of a `t_jrgba` attribute, given its parent object and name.*

### 38.1.1 Detailed Description

An attribute of an object is a setting or property that tells the object how to do its job.

For example, the metro object has an interval attribute that tells it how fast to run.

Attributes are similar to methods, except that the attributes have a state. Attributes are themselves objects, and they share a common interface for getting and setting values.

An attribute is most typically added to the class definition of another object during its class initialization or `ext_main()` function. Most typically, this attribute's value will be stored in an instance's struct, and thus it will serve as a property of that instance of the object.

Attributes can, however, be declared as 'class static'. This means that the property is shared by all instances of the class, and the value is stored as a shared (static) variable.

Additionally, Max 5 has introduced the notion of 'instance attributes' (also called 'object attributes'). Instance attributes are the creation of an attribute object, and then adding it to one specific instance of another class.

Finally, because attributes themselves are Max objects they too can possess attributes. These 'attributes of attributes' are used in Max to do things like specify a range of values for an attribute, give an attribute human friendly caption, or determine to what category an attribute should belong in the inspector.

The easiest and most common way of working with attributes is to use the provided macros. These macros simplify the process of creating a new attribute object, setting any attributes of the attribute, and binding it to an object class or an object instance.

### 38.1.2 Setting and Getting Attribute Values

By default, Max provides standard attribute accessors. These are the functions that get or set the attribute value in the object's struct. If you need to define a custom accessor, you can specify this information using the `CLASS_ATTR_ACCESSORS` macro.

#### 38.1.2.1 Writing a custom Attribute Getter

If you need to define a custom accessor, it should have a prototype and form comparable to the following custom getter:

```
t_max_err foo_myval_get(t_foo *x, void *attr, long *ac, t_atom **av)
{
    if ((*ac)&&(*av)) {
        //memory passed in, use it
    } else {
        //otherwise allocate memory
        *ac = 1;
        if (!(*av = getbytes(sizeof(t_atom)*(*ac)))) {
            *ac = 0;
            return MAX_ERR_OUT_OF_MEM;
        }
    }
    atom_setfloat(*av,x->myval);
    return MAX_ERR_NONE;
}
```

Note that getters require memory to be allocated, if there is no memory passed into the getter. Also the attr argument is the class' attribute object and can be queried using `object_method` for things like the attribute flags, names, filters, etc..

#### 38.1.2.2 Writing a custom Attribute Setter

If you need to define a custom accessor, it should have a prototype and form comparable to the following custom setter:

```
t_max_err foo_myval_set(t_foo *x, void *attr, long ac, t_atom *av)
{
    if (ac&&av) {
        x->myval = atom_getfloat(av);
    } else {
        // no args, set to zero
        x->myval = 0;
    }
    return MAX_ERR_NONE;
}
```

### 38.1.3 Attribute Notificaton

Although the subject of object registration and notification is covered elsewhere, it bears noting that attributes of all types will, if registered, automatically send notifications to all attached client objects each time the attribute's value is set.

### 38.1.4 Macro Definition Documentation

#### 38.1.4.1 CLASS\_ATTR\_ACCESSORS

```
#define CLASS_ATTR_ACCESSORS (
    c,
    attrname,
    getter,
    setter )
```

Specify custom accessor methods for an attribute.

If you specify a non-NULL value for the setter or getter, then the function you specify will be called to set or get the attribute's value rather than using the built-in accessor.

##### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>getter</i>	An appropriate getter method as discussed in <a href="#">Setting and Getting Attribute Values</a> , or NULL to use the default getter.
<i>setter</i>	An appropriate setter method as discussed in <a href="#">Setting and Getting Attribute Values</a> , or NULL to use the default setter.

#### 38.1.4.2 CLASS\_ATTR\_ADD\_FLAGS

```
#define CLASS_ATTR_ADD_FLAGS (
    c,
    attrname,
    flags )
```

Add flags to an attribute.

##### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to add to this attribute, as defined in <a href="#">e_max_attrflags</a> .

### 38.1.4.3 CLASS\_ATTR\_ALIAS

```
#define CLASS_ATTR_ALIAS(
    c,
    attrname,
    aliasname )
```

Create a new attribute that is an alias of an existing attribute.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the actual attribute as a C-string.
<i>aliasname</i>	The name of the new alias attribute.

### 38.1.4.4 CLASS\_ATTR\_ATOM

```
#define CLASS_ATTR_ATOM(
    c,
    attrname,
    flags,
    structname,
    structmember )
```

Create a [t\\_atom](#) attribute and add it to a Max class.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

### 38.1.4.5 CLASS\_ATTR\_ATOM\_ARRAY

```
#define CLASS_ATTR_ATOM_ARRAY(
    c,
```

```
attrname,
flags,
structname,
structmember,
size )
```

Create an array-of-atoms attribute of fixed length, and add it to a Max class.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of items in the <a href="#">t_atom</a> array.

### 38.1.4.6 CLASS\_ATTR\_ATOM\_LONG

```
#define CLASS_ATTR_ATOM_LONG (
    c,
    attrname,
    flags,
    structname,
    structmember )
```

Create a [t\\_atom\\_long](#) integer attribute and add it to a Max class.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

### 38.1.4.7 CLASS\_ATTR\_ATOM\_LONG\_ARRAY

```
#define CLASS_ATTR_ATOM_LONG_ARRAY (
    c,
```

```
attrname,
flags,
structname,
structmember,
size )
```

Create an array-of-t\_atom\_long-integers attribute of fixed length, and add it to a Max class.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of longs in the array.

#### 38.1.4.8 CLASS\_ATTR\_ATOM\_VARSIZE

```
#define CLASS_ATTR_ATOM_VARSIZE (
    c,
    attrname,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-atoms attribute of variable length, and add it to a Max class.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the <a href="#">t_atom</a> array at any given moment.
<i>maxsize</i>	The maximum number of items in the <a href="#">t_atom</a> array, i.e. the number of members allocated for the array in the struct.

### 38.1.4.9 CLASS\_ATTR\_BASIC

```
#define CLASS_ATTR_BASIC(
    c,
    attrname,
    flags )
```

Add a new attribute to the specified attribute to specify that it should appear in the inspector's Basic tab.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .

### 38.1.4.10 CLASS\_ATTR\_CATEGORY

```
#define CLASS_ATTR_CATEGORY(
    c,
    attrname,
    flags,
    parsestr )
```

Add a new attribute to the specified attribute to specify a category to which the attribute is assigned in the Max inspector.

Categories are represented in the inspector as tabs. If the specified category does not exist then it will be created.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

### 38.1.4.11 CLASS\_ATTR\_CHAR

```
#define CLASS_ATTR_CHAR(
    c,
    attrname,
    flags,
    structname,
    structmember )
```

Create a char attribute and add it to a Max class.

## Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

**38.1.4.12 CLASS\_ATTR\_CHAR\_ARRAY**

```
#define CLASS_ATTR_CHAR_ARRAY (
    c,
    attrname,
    flags,
    structname,
    structmember,
    size )
```

Create an array-of-chars attribute of fixed length, and add it to a Max class.

## Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of chars in the array.

**38.1.4.13 CLASS\_ATTR\_CHAR\_VARSIZE**

```
#define CLASS_ATTR_CHAR_VARSIZE (
    c,
    attrname,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-chars attribute of variable length, and add it to a Max class.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the char array at any given moment.
<i>maxsize</i>	The maximum number of items in the char array, i.e. the number of members allocated for the array in the struct.

**38.1.4.14 CLASS\_ATTR\_DEFAULT**

```
#define CLASS_ATTR_DEFAULT(
    c,
    attrname,
    flags,
    parsestr )
```

Add a new attribute to the specified attribute to specify a default value.

The default value will be automatically set when the object is created only if your object uses a dictionary constructor with the [CLASS\\_FLAG\\_NEWDICTIONARY](#) flag.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

**38.1.4.15 CLASS\_ATTR\_DEFAULT\_PAINT**

```
#define CLASS_ATTR_DEFAULT_PAINT(
    c,
    attrname,
    flags,
    parsestr )
```

A convenience wrapper for both [CLASS\\_ATTR\\_DEFAULT](#) and [CLASS\\_ATTR\\_PAINT](#).

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

**See also**

[CLASS\\_ATTR\\_DEFAULT](#)  
[CLASS\\_ATTR\\_PAINT](#)

**38.1.4.16 CLASS\_ATTR\_DEFAULT\_SAVE**

```
#define CLASS_ATTR_DEFAULT_SAVE(
    c,
    attrname,
    flags,
    parsestr )
```

A convenience wrapper for both [CLASS\\_ATTR\\_DEFAULT](#) and [CLASS\\_ATTR\\_SAVE](#).

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

**See also**

[CLASS\\_ATTR\\_DEFAULT](#)  
[CLASS\\_ATTR\\_SAVE](#)

**38.1.4.17 CLASS\_ATTR\_DEFAULT\_SAVE\_PAINT**

```
#define CLASS_ATTR_DEFAULT_SAVE_PAINT(
    c,
    attrname,
    flags,
    parsestr )
```

A convenience wrapper for [CLASS\\_ATTR\\_DEFAULT](#), [CLASS\\_ATTR\\_SAVE](#), and [CLASS\\_ATTR\\_PAINT](#).

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

**See also**

[CLASS\\_ATTR\\_DEFAULT](#)  
[CLASS\\_ATTR\\_PAINT](#)  
[CLASS\\_ATTR\\_SAVE](#)

**38.1.4.18 CLASS\_ATTR\_DEFAULTNAME**

```
#define CLASS_ATTR_DEFAULTNAME (
    c,
    attrname,
    flags,
    parsestr )
```

Add a new attribute to the specified attribute to specify a default value, based on Max's Object Defaults.

If a value is present in Max's Object Defaults, then that value will be used as the default value. Otherwise, use the default value specified here. The default value will be automatically set when the object is created only if your object uses a dictionary constructor with the [CLASS\\_FLAG\\_NEWDICTIONARY](#) flag.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

**38.1.4.19 CLASS\_ATTR\_DEFAULTNAME\_PAINT**

```
#define CLASS_ATTR_DEFAULTNAME_PAINT (
    c,
    attrname,
    flags,
    parsestr )
```

A convenience wrapper for [CLASS\\_ATTR\\_DEFAULTNAME](#), [CLASS\\_ATTR\\_SAVE](#), and [CLASS\\_ATTR\\_PAINT](#).

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

**See also**

[CLASS\\_ATTR\\_DEFAULTNAME](#)  
[CLASS\\_ATTR\\_PAINT](#)  
[CLASS\\_ATTR\\_SAVE](#)

**38.1.4.20 CLASS\_ATTR\_DEFAULTNAME\_SAVE**

```
#define CLASS_ATTR_DEFAULTNAME_SAVE (
    c,
    attrname,
    flags,
    parsestr )
```

A convenience wrapper for both [CLASS\\_ATTR\\_DEFAULTNAME](#) and [CLASS\\_ATTR\\_SAVE](#).

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

**See also**

[CLASS\\_ATTR\\_DEFAULTNAME](#)  
[CLASS\\_ATTR\\_SAVE](#)

**38.1.4.21 CLASS\_ATTR\_DEFAULTNAME\_SAVE\_PAINT**

```
#define CLASS_ATTR_DEFAULTNAME_SAVE_PAINT (
    c,
    attrname,
```

```
flags,  
parsestr )
```

A convenience wrapper for [CLASS\\_ATTR\\_DEFAULTNAME](#), [CLASS\\_ATTR\\_SAVE](#), and [CLASS\\_ATTR\\_PAINT](#).

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

**See also**

[CLASS\\_ATTR\\_DEFAULTNAME](#)  
[CLASS\\_ATTR\\_PAINT](#)  
[CLASS\\_ATTR\\_SAVE](#)

**38.1.4.22 CLASS\_ATTR\_DOUBLE**

```
#define CLASS_ATTR_DOUBLE(
    c,
    attrname,
    flags,
    structname,
    structmember )
```

Create a 64-bit float attribute and add it to a Max class.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

**38.1.4.23 CLASS\_ATTR\_DOUBLE\_ARRAY**

```
#define CLASS_ATTR_DOUBLE_ARRAY(
    c,
    attrname,
    flags,
    structname,
```

```
structmember,  
size )
```

Create an array-of-64bit-floats attribute of fixed length, and add it to a Max class.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of doubles in the array.

**38.1.4.24 CLASS\_ATTR\_DOUBLE\_VARSIZE**

```
#define CLASS_ATTR_DOUBLE_VARSIZE(
    c,
    attrname,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-64bit-floats attribute of variable length, and add it to a Max class.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the double array at any given moment.
<i>maxsize</i>	The maximum number of items in the double array, i.e. the number of members allocated for the array in the struct.

**38.1.4.25 CLASS\_ATTR\_ENUM**

```
#define CLASS_ATTR_ENUM(
    c,
    attrname,
    flags,
    parsestr )
```

Add a new attribute to the specified attribute to specify a list of choices to display in a menu for the Max inspector.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

**Remarks**

This macro automatically calls  
`CLASS_ATTR_STYLE(c, attrname, flags, "enum") .`

**See also**

[CLASS\\_ATTR\\_ENUMINDEX](#)

**38.1.4.26 CLASS\_ATTR\_ENUMINDEX**

```
#define CLASS_ATTR_ENUMINDEX(
    c,
    attrname,
    flags,
    parsestr )
```

Add a new attribute to the specified attribute to specify a list of choices to display in a menu for the Max inspector.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

**Remarks**

This macro automatically calls  
`CLASS_ATTR_STYLE(c, attrname, flags, "enumindex") .`

**See also**

[CLASS\\_ATTR\\_ENUM](#)

### 38.1.4.27 CLASS\_ATTR\_FILTER\_CLIP

```
#define CLASS_ATTR_FILTER_CLIP(
    c,
    attrname,
    minval,
    maxval )
```

Add a filter to the attribute to limit both the lower and upper bounds of a value.

The limiting will be performed by the default attribute accessor.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>minval</i>	The maximum acceptable value to which the attribute will be limited.
<i>maxval</i>	The maximum acceptable value to which the attribute will be limited.

#### See also

### 38.1.4.28 CLASS\_ATTR\_FILTER\_MAX

```
#define CLASS_ATTR_FILTER_MAX(
    c,
    attrname,
    maxval )
```

Add a filter to the attribute to limit the upper bound of a value.

The limiting will be performed by the default attribute accessor.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>maxval</i>	The maximum acceptable value to which the attribute will be limited.

#### See also

[CLASS\\_ATTR\\_FILTER\\_MIN](#)  
[CLASS\\_ATTR\\_FILTER\\_CLIP](#)  
[CLASS\\_ATTR\\_MAX](#)

### 38.1.4.29 CLASS\_ATTR\_FILTER\_MIN

```
#define CLASS_ATTR_FILTER_MIN(
    c,
    attrname,
    minval )
```

Add a filter to the attribute to limit the lower bound of a value.

The limiting will be performed by the default attribute accessor.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>minval</i>	The minimum acceptable value to which the attribute will be limited.

#### See also

[CLASS\\_ATTR\\_FILTER\\_MAX](#)  
[CLASS\\_ATTR\\_FILTER\\_CLIP](#)  
[CLASS\\_ATTR\\_MIN](#)

### 38.1.4.30 CLASS\_ATTR\_FLOAT

```
#define CLASS_ATTR_FLOAT(
    c,
    attrname,
    flags,
    structname,
    structmember )
```

Create a 32-bit float attribute and add it to a Max class.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

### 38.1.4.31 CLASS\_ATTR\_FLOAT\_ARRAY

```
#define CLASS_ATTR_FLOAT_ARRAY(
    c,
    attrname,
    flags,
    structname,
    structmember,
    size )
```

Create an array-of-32bit-floats attribute of fixed length, and add it to a Max class.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of floats in the array.

### 38.1.4.32 CLASS\_ATTR\_FLOAT\_VARSIZE

```
#define CLASS_ATTR_FLOAT_VARSIZE(
    c,
    attrname,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-32bit-floats attribute of variable length, and add it to a Max class.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the float array at any given moment.
<i>maxsize</i>	The maximum number of items in the float array, i.e. the number of members allocated for the array in the struct.

### 38.1.4.33 CLASS\_ATTR\_INT32

```
#define CLASS_ATTR_INT32 (
    c,
    attrname,
    flags,
    structname,
    structmember )
```

Create a `t_int32` integer attribute and add it to a Max class.

#### Parameters

<code>c</code>	The class pointer.
<code>attrname</code>	The name of this attribute as a C-string.
<code>flags</code>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<code>structname</code>	The C identifier for the struct (containing a valid <code>t_object</code> header) representing an instance of this class.
<code>structmember</code>	The C identifier of the member in the struct that holds the value of this attribute.

### 38.1.4.34 CLASS\_ATTR\_INTRODUCED

```
#define CLASS_ATTR_INTRODUCED (
    c,
    attrname,
    flags,
    versionstr )
```

Add a new attribute to the specified attribute to indicate in which version the attribute was introduced.

#### Parameters

<code>c</code>	The class pointer.
<code>attrname</code>	The name of the attribute as a C-string.
<code>flags</code>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<code>versionstr</code>	A C-string, which will be parsed set the version number (e.g. "7.0.0").

### 38.1.4.35 CLASS\_ATTR\_INVISIBLE

```
#define CLASS_ATTR_INVISIBLE (
```

```
c,
attrname,
flags )
```

Add a new attribute to the specified attribute to flag an attribute as invisible to the Max inspector.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .

### 38.1.4.36 CLASS\_ATTR\_LABEL

```
#define CLASS_ATTR_LABEL(
    c,
    attrname,
    flags,
    labelstr )
```

Add a new attribute to the specified attribute to specify an a human-friendly label for the Max inspector.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>labelstr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

### 38.1.4.37 CLASS\_ATTR\_LEGACYDEFAULT

```
#define CLASS_ATTR_LEGACYDEFAULT(
    c,
    legacyattrname,
    newattrname,
    flags,
    parsestr )
```

Add a new attribute to the specified attribute to specify a legacy default value.

The default value will be automatically set when the object is created only if your object uses a dictionary constructor with the [CLASS\\_FLAG\\_NEWDICTIONARY](#) flag.

**Parameters**

<i>c</i>	The class pointer.
<i>legacyattrname</i>	The name of the attribute.
<i>newattrname</i>	The name of the attribute.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the legacy value, used by <code>jbox_processlegacydefaults()</code>

**38.1.4.38 CLASS\_ATTR\_LONG**

```
#define CLASS_ATTR_LONG(
    c,
    attrname,
    flags,
    structname,
    structmember )
```

Create a long integer attribute and add it to a Max class.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

**38.1.4.39 CLASS\_ATTR\_LONG\_ARRAY**

```
#define CLASS_ATTR_LONG_ARRAY(
    c,
    attrname,
    flags,
    structname,
    structmember,
    size )
```

Create an array-of-long-integers attribute of fixed length, and add it to a Max class.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of longs in the array.

**38.1.4.40 CLASS\_ATTR\_LONG\_VARSIZE**

```
#define CLASS_ATTR_LONG_VARSIZE (
    c,
    attrname,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-long-integers attribute of variable length, and add it to a Max class.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the long array at any given moment.
<i>maxsize</i>	The maximum number of items in the long array, i.e. the number of members allocated for the array in the struct.

**38.1.4.41 CLASS\_ATTR\_MAX**

```
#define CLASS_ATTR_MAX (
    c,
    attrname,
    flags,
    parsestr )
```

Add a new attribute to the specified attribute to specify an upper range.

The values will not be automatically limited.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

#### See also

[CLASS\\_ATTR\\_MIN](#)  
[CLASS\\_ATTR\\_FILTER\\_MAX](#)  
[CLASS\\_ATTR\\_FILTER\\_CLIP](#)

#### 38.1.4.42 CLASS\_ATTR\_MIN

```
#define CLASS_ATTR_MIN(
    c,
    attrname,
    flags,
    parsestr )
```

Add a new attribute to the specified attribute to specify a lower range.

The values will not be automatically limited.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

#### See also

[CLASS\\_ATTR\\_MAX](#)  
[CLASS\\_ATTR\\_FILTER\\_MAX](#)  
[CLASS\\_ATTR\\_FILTER\\_CLIP](#)

### 38.1.4.43 CLASS\_ATTR\_OBJ

```
#define CLASS_ATTR_OBJ(
    c,
    attrname,
    flags,
    structname,
    structmember )
```

Create a [t\\_object\\*](#) attribute and add it to a Max class.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

### 38.1.4.44 CLASS\_ATTR\_OBJ\_ARRAY

```
#define CLASS_ATTR_OBJ_ARRAY(
    c,
    attrname,
    flags,
    structname,
    structmember,
    size )
```

Create an array-of-objects attribute of fixed length, and add it to a Max class.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of items in the <a href="#">t_object*</a> array.

### 38.1.4.45 CLASS\_ATTR\_OBJ\_VARSIZE

```
#define CLASS_ATTR_OBJ_VARSIZE(
    c,
    attrname,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-objects attribute of variable length, and add it to a Max class.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the <a href="#">t_object</a> * array at any given moment.
<i>maxsize</i>	The maximum number of items in the <a href="#">t_object</a> * array, i.e. the number of members allocated for the array in the struct.

### 38.1.4.46 CLASS\_ATTR\_OBSOLETE

```
#define CLASS_ATTR_OBSOLETE(
    c,
    attrname,
    flags )
```

Add a new attribute to the specified attribute to flag it as obsolete.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .

### 38.1.4.47 CLASS\_ATTR\_OFFSET\_DUMMY

```
#define CLASS_ATTR_OFFSET_DUMMY(
```

```
c,
attrname,
flags,
typesym )
```

Create an attribute that does not store its data in the object struct.

NB: if you use this you must have a custom getter/setter or not ever get/set. Perhaps we should rewrite this using a generic attribute\_new rather than attr\_offset\_new?

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>typesym</i>	The type the getter and setter would expect: _sym_char, _sym_long, _sym_atom_long, _sym_float32, _sym_float64, _sym_symbol, _sym_atom, etc

#### 38.1.4.48 CLASS\_ATTR\_ORDER

```
#define CLASS_ATTR_ORDER (
    c,
    attrname,
    flags,
    parsestr )
```

Add a new attribute to the specified attribute to specify a default order in which to list attributes.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

#### Remarks

A value of zero indicates that there is no ordering. Ordering values begin at 1. For example:

```
CLASS_ATTR_ORDER(c, "firstattr", 0, "1");
CLASS_ATTR_ORDER(c, "secondattr", 0, "2");
CLASS_ATTR_ORDER(c, "thirdattr", 0, "3");
```

### 38.1.4.49 CLASS\_ATTR\_PAINT

```
#define CLASS_ATTR_PAINT(
    c,
    attrname,
    flags )
```

Add a new attribute indicating that any changes to the specified attribute will trigger a call to the object's paint method.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .

### 38.1.4.50 CLASS\_ATTR\_REMOVE\_FLAGS

```
#define CLASS_ATTR_REMOVE_FLAGS(
    c,
    attrname,
    flags )
```

Remove flags from an attribute.

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to remove from this attribute, as defined in <a href="#">e_max_attrflags</a> .

### 38.1.4.51 CLASS\_ATTR\_RENAMED

```
#define CLASS_ATTR_RENAMED(
    c,
    oldname,
    newname,
    flags )
```

Add a new attribute to the specified attribute to flag it as renamed.

**Parameters**

<i>c</i>	The class pointer.
<i>oldname</i>	The name of the old attribute as a C-string.
<i>newname</i>	The name of the new attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .

**38.1.4.52 CLASS\_ATTR\_RGBA**

```
#define CLASS_ATTR_RGBA(
    c,
    attrname,
    flags,
    structname,
    structmember )
```

Create a color ([t\\_jrgba](#)) attribute and add it to a Max class.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

**38.1.4.53 CLASS\_ATTR\_SAVE**

```
#define CLASS_ATTR_SAVE(
    c,
    attrname,
    flags )
```

Add a new attribute to the specified attribute to indicate that the specified attribute should be saved with the patcher.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .

### 38.1.4.54 CLASS\_ATTR\_SELFSAVE

```
#define CLASS_ATTR_SELFSAVE (
    c,
    attrname,
    flags )
```

Add a new attribute to the specified attribute to indicate that it is saved by the object (so it does not appear in italics in the inspector).

#### Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .

### 38.1.4.55 CLASS\_ATTR\_STYLE

```
#define CLASS_ATTR_STYLE (
    c,
    attrname,
    flags,
    parsestr )
```

Add a new attribute to the specified attribute to specify an editor style for the Max inspector.

Available styles include

- "text" : a text editor
- "onoff" : a toggle switch
- "rgba" : a color chooser
- "enum" : a menu of available choices, whose symbol will be passed upon selection
- "enumindex" : a menu of available choices, whose index will be passed upon selection
- "rect" : a style for displaying and editing [t\\_rect](#) values
- "font" : a font chooser
- "file" : a file chooser dialog

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

**38.1.4.56 CLASS\_ATTR\_STYLE\_LABEL**

```
#define CLASS_ATTR_STYLE_LABEL(
    c,
    attrname,
    flags,
    stylestr,
    labelstr )
```

A convenience wrapper for [CLASS\\_ATTR\\_STYLE](#), and [CLASS\\_ATTR\\_LABEL](#).

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>stylestr</i>	A C-string that names the style for the attribute. See <a href="#">CLASS_ATTR_STYLE</a> for the available styles.
<i>labelstr</i>	A C-string that names the category to which the attribute is assigned in the inspector.

**See also**

[CLASS\\_ATTR\\_STYLE](#)  
[CLASS\\_ATTR\\_LABEL](#)

**38.1.4.57 CLASS\_ATTR\_SYM**

```
#define CLASS_ATTR_SYM(
    c,
    attrname,
    flags,
    structname,
    structmember )
```

Create a [t\\_symbol\\*](#) attribute and add it to a Max class.

## Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

**38.1.4.58 CLASS\_ATTR\_SYM\_ARRAY**

```
#define CLASS_ATTR_SYM_ARRAY(
    c,
    attrname,
    flags,
    structname,
    structmember,
    size )
```

Create an array-of-symbols attribute of fixed length, and add it to a Max class.

## Parameters

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of items in the <a href="#">t_symbol</a> * array.

**38.1.4.59 CLASS\_ATTR\_SYM\_VARSIZE**

```
#define CLASS_ATTR_SYM_VARSIZE(
    c,
    attrname,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-symbols attribute of variable length, and add it to a Max class.

**Parameters**

<i>c</i>	The class pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the <a href="#">t_symbol</a> * array at any given moment.
<i>maxsize</i>	The maximum number of items in the <a href="#">t_symbol</a> * array, i.e. the number of members allocated for the array in the struct.

**38.1.4.60 CLASS\_METHOD\_ATTR\_PARSE**

```
#define CLASS_METHOD_ATTR_PARSE(
    c,
    methodname,
    attrname,
    type,
    flags,
    parsestring )
```

Define and add attributes to class methods.

**Parameters**

<i>c</i>	The class pointer.
<i>methodname</i>	The name of the existing method as a C-string.
<i>attrname</i>	The name of the attribute to add as a C-string.
<i>type</i>	The datatype of the attribute to be added.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestring</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

**Remarks**

An example which makes a method invisible to users:

```
class_admmethod(c, (method)my_foo, "foo", 0);
CLASS_METHOD_ATTR_PARSE(c, "foo", "undocumented", gensym("long"), 0, "1");
```

**38.1.4.61 CLASS\_METHOD\_INTRODUCED**

```
#define CLASS_METHOD_INTRODUCED(
    c,
```

```
methodname,
flags,
versionstr )
```

Add a new attribute to the specified method to indicate in which version the method was introduced.

#### Parameters

<i>c</i>	The class pointer.
<i>methodname</i>	The name of the method as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>versionstr</i>	A C-string, which will be parsed set the version number (e.g. "7.0.0").

#### 38.1.4.62 CLASS\_METHOD\_OBSOLETE

```
#define CLASS_METHOD_OBSOLETE (
    c,
    methodname,
    flags )
```

Add a new attribute to the specified method to flag it as obsolete.

#### Parameters

<i>c</i>	The class pointer.
<i>methodname</i>	The name of the method as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .

#### 38.1.4.63 CLASS\_METHOD\_RENAMED

```
#define CLASS_METHOD_RENAMED (
    c,
    oldname,
    newname,
    flags )
```

Add a new attribute to the specified method to flag a method as renamed.

#### Parameters

<i>c</i>	The class pointer.
<i>oldname</i>	The name of the old method as a C-string.
<i>newname</i>	The name of the new method as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .

cycling 74

### 38.1.4.64 CLASS\_STICKY\_ATTR

```
#define CLASS_STICKY_ATTR(
    c,
    name,
    flags,
    parsestr )
```

Create an attribute, and add it to all following attribute declarations.

The block is closed by a call to [CLASS\\_STICKY\\_ATTR\\_CLEAR](#).

#### Parameters

<i>c</i>	The class pointer.
<i>name</i>	The name of the new attribute to create as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

#### Remarks

The most common use of CLASS\_STICKY\_ATTR is for creating multiple attributes with the same category, as in this example:

```
CLASS_STICKY_ATTR(c, "category", 0, "Foo");
CLASS_ATTR_DOUBLE(c, "bar", 0, t_myobject, x_bar);
CLASS_ATTR_LABEL(c, "bar", 0, "A Bar");
CLASS_ATTR_CHAR(c, "switch", 0, t_myobject, x_switch);
CLASS_ATTR_STYLE_LABEL(c, "switch", 0, "onoff", "Bar Switch");
CLASS_ATTR_DOUBLE(c, "flow", 0, t_myobject, x_flow);
CLASS_ATTR_LABEL(c, "flow", 0, "Flow Amount");
CLASS_STICKY_ATTR_CLEAR(c, "category");
```

#### See also

[CLASS\\_STICKY\\_ATTR\\_CLEAR](#)

### 38.1.4.65 CLASS\_STICKY\_ATTR\_CLEAR

```
#define CLASS_STICKY_ATTR_CLEAR(
    c,
    name )
```

Close a [CLASS\\_STICKY\\_ATTR](#) block.

#### Parameters

<i>c</i>	The class pointer.
<i>name</i>	The name of the sticky attribute as a C-string.

#### See also

[CLASS\\_STICKY\\_ATTR](#)

#### 38.1.4.66 CLASS\_STICKY\_METHOD

```
#define CLASS_STICKY_METHOD (
    c,
    name,
    flags,
    parsestr )
```

Create an attribute, and add it to all following method declarations.

The block is closed by a call to [CLASS\\_STICKY\\_METHOD\\_CLEAR](#).

#### Parameters

<i>c</i>	The class pointer.
<i>name</i>	The name of the new attribute to create as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

#### Remarks

The most common use of CLASS\_STICKY\_ATTR is for creating multiple attributes with the same category, as in this example:

```
CLASS_STICKY_METHOD(c, "undocumented", 0, "1");
// add some methods here with class_adddmethod()
// the undocumented attribute for methods means that the ref-page
// generator will ignore these methods.
CLASS_STICKY_METHOD_CLEAR(c, "undocumented");
```

#### See also

[CLASS\\_STICKY\\_METHOD\\_CLEAR](#)

### 38.1.4.67 CLASS\_STICKY\_METHOD\_CLEAR

```
#define CLASS_STICKY_METHOD_CLEAR(
    c,
    name )
```

Close a [CLASS\\_STICKY\\_METHOD](#) block.

#### Parameters

<i>c</i>	The class pointer.
<i>name</i>	The name of the sticky attribute as a C-string.

#### See also

[CLASS\\_STICKY\\_METHOD](#)

### 38.1.4.68 OBJ\_ATTR\_ATOM

```
#define OBJ_ATTR_ATOM(
    x,
    attrname,
    flags,
    val )
```

Create an instance-local [t\\_atom](#) attribute and add it to a Max class.

#### Parameters

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>val</i>	Pointer to the value.

### 38.1.4.69 OBJ\_ATTR\_ATOM\_ARRAY

```
#define OBJ_ATTR_ATOM_ARRAY
```

Create an instance-local array-of-atoms attribute of fixed length, and add it to the object.

**Parameters**

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>count</i>	The number of items in the <a href="#">t_atom</a> array.
<i>vals</i>	Pointer to the values.

**38.1.4.70 OBJ\_ATTR\_CHAR**

```
#define OBJ_ATTR_CHAR(
    x,
    attrname,
    flags,
    val )
```

Create an instance-local char attribute and add it to a Max class.

**Parameters**

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>val</i>	Pointer to the value.

**38.1.4.71 OBJ\_ATTR\_CHAR\_ARRAY**

```
#define OBJ_ATTR_CHAR_ARRAY(
    x,
    attrname,
    flags,
    count,
    vals )
```

Create an instance-local array-of-chars attribute of fixed length, and add it to the object.

**Parameters**

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>count</i>	The number of items in the char array.
<i>vals</i>	Pointer to the values.

### 38.1.4.72 OBJ\_ATTR\_DEFAULT

```
#define OBJ_ATTR_DEFAULT (
    x,
    attrname,
    flags,
    parsestr )
```

An instance-attribute version of [CLASS\\_ATTR\\_DEFAULT](#).

#### Parameters

<i>x</i>	The <a href="#">t_object</a> instance pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

#### See also

[CLASS\\_ATTR\\_DEFAULT](#)

### 38.1.4.73 OBJ\_ATTR\_DEFAULT\_SAVE

```
#define OBJ_ATTR_DEFAULT_SAVE (
    x,
    attrname,
    flags,
    parsestr )
```

An instance-attribute version of [CLASS\\_ATTR\\_DEFAULT\\_SAVE](#).

#### Parameters

<i>x</i>	The <a href="#">t_object</a> instance pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>parsestr</i>	A C-string, which will be parsed into an array of atoms to set the initial value.

#### See also

[CLASS\\_ATTR\\_DEFAULT\\_SAVE](#)

### 38.1.4.74 OBJ\_ATTR\_DOUBLE

```
#define OBJ_ATTR_DOUBLE (
    x,
    attrname,
    flags,
    val )
```

Create an instance-local 64bit float attribute and add it to a Max class.

#### Parameters

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>val</i>	Pointer to the value.

### 38.1.4.75 OBJ\_ATTR\_DOUBLE\_ARRAY

```
#define OBJ_ATTR_DOUBLE_ARRAY (
    x,
    attrname,
    flags,
    count,
    vals )
```

Create an instance-local array-of-64bit-floats attribute of fixed length, and add it to the object.

#### Parameters

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>count</i>	The number of items in the double array.
<i>vals</i>	Pointer to the values.

### 38.1.4.76 OBJ\_ATTR\_FLOAT

```
#define OBJ_ATTR_FLOAT (
    x,
    attrname,
```

```
flags,
val )
```

Create an instance-local 32bit float attribute and add it to a Max class.

#### Parameters

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>val</i>	Pointer to the value.

### 38.1.4.77 OBJ\_ATTR\_FLOAT\_ARRAY

```
#define OBJ_ATTR_FLOAT_ARRAY (
    x,
    attrname,
    flags,
    count,
    vals )
```

Create an instance-local array-of-32bit-floats attribute of fixed length, and add it to the object.

#### Parameters

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>count</i>	The number of items in the float array.
<i>vals</i>	Pointer to the values.

### 38.1.4.78 OBJ\_ATTR\_LONG

```
#define OBJ_ATTR_LONG (
    x,
    attrname,
    flags,
    val )
```

Create an instance-local long integer attribute and add it to a Max class.

## Parameters

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>val</i>	Pointer to the value.

**38.1.4.79 OBJ\_ATTR\_LONG\_ARRAY**

```
#define OBJ_ATTR_LONG_ARRAY(
    x,
    attrname,
    flags,
    count,
    vals )
```

Create an instance-local array-of-long-integers attribute of fixed length, and add it to the object.

## Parameters

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>count</i>	The number of items in the long array.
<i>vals</i>	Pointer to the values.

**38.1.4.80 OBJ\_ATTR\_OBJ**

```
#define OBJ_ATTR_OBJ(
    x,
    attrname,
    flags,
    val )
```

Create an instance-local [t\\_object](#)\* attribute and add it to a Max class.

## Parameters

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>val</i>	Pointer to the value.

### 38.1.4.81 OBJ\_ATTR\_OBJ\_ARRAY

```
#define OBJ_ATTR_OBJ_ARRAY(
    x,
    attrname,
    flags,
    count,
    vals )
```

Create an instance-local array-of-objects attribute of fixed length, and add it to the object.

#### Parameters

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>count</i>	The number of items in the <a href="#">t_object</a> * array.
<i>vals</i>	Pointer to the values.

### 38.1.4.82 OBJ\_ATTR\_SAVE

```
#define OBJ_ATTR_SAVE(
    x,
    attrname,
    flags )
```

An instance-attribute version of [CLASS\\_ATTR\\_SAVE](#).

#### Parameters

<i>x</i>	The <a href="#">t_object</a> instance pointer.
<i>attrname</i>	The name of the attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this new attribute, as defined in <a href="#">e_max_attrflags</a> .

#### See also

[CLASS\\_ATTR\\_SAVE](#)

### 38.1.4.83 OBJ\_ATTR\_SYM

```
#define OBJ_ATTR_SYM(
    x,
    attrname,
    flags,
    val )
```

Create an instance-local [t\\_symbol\\*](#) attribute and add it to a Max class.

#### Parameters

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>val</i>	Pointer to the value.

### 38.1.4.84 OBJ\_ATTR\_SYM\_ARRAY

```
#define OBJ_ATTR_SYM_ARRAY(
    x,
    attrname,
    flags,
    count,
    vals )
```

Create an instance-local array-of-symbols attribute of fixed length, and add it to the object.

#### Parameters

<i>x</i>	The object pointer.
<i>attrname</i>	The name of this attribute as a C-string.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>count</i>	The number of items in the <a href="#">t_symbol*</a> array.
<i>vals</i>	Pointer to the values.

### 38.1.4.85 STRUCT\_ATTR\_ATOM

```
#define STRUCT_ATTR_ATOM(
    c,
    flags,
```

```
structname,
structmember )
```

Create a [t\\_atom](#) attribute and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

### 38.1.4.86 STRUCT\_ATTR\_ATOM\_ARRAY

```
#define STRUCT_ATTR_ATOM_ARRAY(
    c,
    flags,
    structname,
    structmember,
    size )
```

Create an array-of-atoms attribute of fixed length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of items in the <a href="#">t_atom</a> array.

### 38.1.4.87 STRUCT\_ATTR\_ATOM\_LONG

```
#define STRUCT_ATTR_ATOM_LONG(
    c,
    flags,
```

```
structname,
structmember )
```

Create a `t_atom_long` integer attribute and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <code>t_object</code> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

#### 38.1.4.88 STRUCT\_ATTR\_ATOM\_VARSIZE

```
#define STRUCT_ATTR_ATOM_VARSIZE(
    c,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-atoms attribute of variable length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <code>t_object</code> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the <code>t_atom</code> array at any given moment.
<i>maxsize</i>	The maximum number of items in the <code>t_atom</code> array, i.e. the number of members allocated for the array in the struct.

#### 38.1.4.89 STRUCT\_ATTR\_CHAR

```
#define STRUCT_ATTR_CHAR (
    c,
```

```
flags,
structname,
structmember )
```

Create a char attribute and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

### 38.1.4.90 STRUCT\_ATTR\_CHAR\_ARRAY

```
#define STRUCT_ATTR_CHAR_ARRAY(
    c,
    flags,
    structname,
    structmember,
    size )
```

Create an array-of-chars attribute of fixed length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of items in the char array.

### 38.1.4.91 STRUCT\_ATTR\_CHAR\_VARSIZE

```
#define STRUCT_ATTR_CHAR_VARSIZE(
    c,
```

```
flags,
structname,
structmember,
sizemember,
maxsize )
```

Create an array-of-chars attribute of variable length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the char array at any given moment.
<i>maxsize</i>	The maximum number of items in the char array, i.e. the number of members allocated for the array in the struct.

#### 38.1.4.92 STRUCT\_ATTR\_DOUBLE

```
#define STRUCT_ATTR_DOUBLE(
    c,
    flags,
    structname,
    structmember )
```

Create a 64bit float attribute and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

#### 38.1.4.93 STRUCT\_ATTR\_DOUBLE\_ARRAY

```
#define STRUCT_ATTR_DOUBLE_ARRAY(
```

```
c,
flags,
structname,
structmember,
size )
```

Create an array-of-64bit-floats attribute of fixed length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of items in the double array.

### 38.1.4.94 STRUCT\_ATTR\_DOUBLE\_VARSIZE

```
#define STRUCT_ATTR_DOUBLE_VARSIZE(
    c,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-64bit-floats attribute of variable length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the double array at any given moment.
<i>maxsize</i>	The maximum number of items in the double array, i.e. the number of members allocated for the array in the struct.

### 38.1.4.95 STRUCT\_ATTR\_FLOAT

```
#define STRUCT_ATTR_FLOAT(
    c,
    flags,
    structname,
    structmember )
```

Create a 32bit float attribute and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

### 38.1.4.96 STRUCT\_ATTR\_FLOAT\_ARRAY

```
#define STRUCT_ATTR_FLOAT_ARRAY(
    c,
    flags,
    structname,
    structmember,
    size )
```

Create an array-of-32bit-floats attribute of fixed length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of items in the floats array.

### 38.1.4.97 STRUCT\_ATTR\_FLOAT\_VARSIZE

```
#define STRUCT_ATTR_FLOAT_VARSIZE(
    c,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-32bit-floats attribute of variable length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the float array at any given moment.
<i>maxsize</i>	The maximum number of items in the float array, i.e. the number of members allocated for the array in the struct.

### 38.1.4.98 STRUCT\_ATTR\_LONG

```
#define STRUCT_ATTR_LONG(
    c,
    flags,
    structname,
    structmember )
```

Create a long integer attribute and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

### 38.1.4.99 STRUCT\_ATTR\_LONG\_ARRAY

```
#define STRUCT_ATTR_LONG_ARRAY(
    c,
    flags,
    structname,
    structmember,
    size )
```

Create an array-of-long-integers attribute of fixed length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of items in the long array.

### 38.1.4.100 STRUCT\_ATTR\_LONG\_VARSIZE

```
#define STRUCT_ATTR_LONG_VARSIZE(
    c,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-long-integers attribute of variable length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the long array at any given moment.
<i>maxsize</i>	The maximum number of items in the long array, i.e. the number of members allocated for the array in the struct.

### 38.1.4.101 STRUCT\_ATTR\_OBJ

```
#define STRUCT_ATTR_OBJ(
    c,
    flags,
    structname,
    structmember )
```

Create a [t\\_object\\*](#) attribute and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

### 38.1.4.102 STRUCT\_ATTR\_OBJ\_ARRAY

```
#define STRUCT_ATTR_OBJ_ARRAY(
    c,
    flags,
    structname,
    structmember,
    size )
```

Create an array-of-objects attribute of fixed length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of items in the <a href="#">t_object*</a> array.

### 38.1.4.103 STRUCT\_ATTR\_OBJ\_VARSIZE

```
#define STRUCT_ATTR_OBJ_VARSIZE(
    c,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-objects attribute of variable length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the <a href="#">t_object</a> * array at any given moment.
<i>maxsize</i>	The maximum number of items in the <a href="#">t_object</a> * array, i.e. the number of members allocated for the array in the struct.

### 38.1.4.104 STRUCT\_ATTR\_SYM

```
#define STRUCT_ATTR_SYM(
    c,
    flags,
    structname,
    structmember )
```

Create a [t\\_symbol](#)\* attribute and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.

### 38.1.4.105 STRUCT\_ATTR\_SYM\_ARRAY

```
#define STRUCT_ATTR_SYM_ARRAY(
    c,
    flags,
    structname,
    structmember,
    size )
```

Create an array-of-symbols attribute of fixed length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>size</i>	The number of items in the <a href="#">t_symbol</a> * array.

### 38.1.4.106 STRUCT\_ATTR\_SYM\_VARSIZE

```
#define STRUCT_ATTR_SYM_VARSIZE(
    c,
    flags,
    structname,
    structmember,
    sizemember,
    maxsize )
```

Create an array-of-symbols attribute of variable length, and add it to a Max class.

The name of the attribute is automatically determined by the name of the struct member.

#### Parameters

<i>c</i>	The class pointer.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>sizemember</i>	The actual number of items in the <a href="#">t_symbol</a> * array at any given moment.
<i>maxsize</i>	The maximum number of items in the <a href="#">t_symbol</a> * array, i.e. the number of members allocated for the array in the struct.

### 38.1.5 Enumeration Type Documentation

#### 38.1.5.1 e\_max\_attrflags

```
enum e_max_attrflags
```

Attribute flags.

##### Remarks

To create a readonly attribute, for example, you should pass ATTR\_SET\_OPAQUE or ATTR\_SET\_OPAQUE\_USER as a flag when you create your attribute.

##### Enumerator

ATTR_FLAGS_NONE	No flags.
ATTR_GET_OPAQUE	The attribute cannot be queried by either max message when used inside of a CLASS_BOX object, nor from C code.
ATTR_SET_OPAQUE	The attribute cannot be set by either max message when used inside of a CLASS_BOX object, nor from C code.
ATTR_GET_OPAQUE_USER	The attribute cannot be queried by max message when used inside of a CLASS_BOX object, but <i>can</i> be queried from C code.
ATTR_SET_OPAQUE_USER	The attribute cannot be set by max message when used inside of a CLASS_BOX object, but <i>can</i> be set from C code.

### 38.1.6 Function Documentation

#### 38.1.6.1 attr\_addfilter\_clip()

```
t_max_err attr_addfilter_clip (
    void * x,
    double min,
    double max,
    long usemin,
    long usemax )
```

Attaches a clip filter to an attribute.

The filter will clip any values sent to or retrieved from the attribute using the attribute's get and set functions.

**Parameters**

<i>x</i>	Pointer to the attribute to receive the filter
<i>min</i>	Minimum value for the clip filter
<i>max</i>	Maximum value for the clip filter
<i>usemin</i>	Sets this value to 0 if the minimum clip value should <i>not</i> be used. Otherwise, set the value to non-zero.
<i>usemax</i>	Sets this value to 0 if the maximum clip value should <i>not</i> be used. Otherwise, set the value to non-zero.

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

**38.1.6.2 attr\_addfilter\_clip\_scale()**

```
t_max_err attr_addfilter_clip_scale (
    void * x,
    double scale,
    double min,
    double max,
    long usemin,
    long usemax )
```

Attaches a clip/scale filter to an attribute.

The filter will clip and scale any values sent to or retrieved from the attribute using the attribute's *get* and *set* functions.

**Parameters**

<i>x</i>	Pointer to the attribute to receive the filter
<i>scale</i>	Scale value. Data sent to the attribute will be scaled by this amount. Data retrieved from the attribute will be scaled by its reciprocal. <i>Scaling occurs previous to clipping.</i>
<i>min</i>	Minimum value for the clip filter
<i>max</i>	Maximum value for the clip filter
<i>usemin</i>	Sets this value to 0 if the minimum clip value should <i>not</i> be used. Otherwise, set the value to non-zero.
<i>usemax</i>	Sets this value to 0 if the maximum clip value should <i>not</i> be used. Otherwise, set the value to non-zero.

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

### 38.1.6.3 attr\_addfilterget\_clip()

```
t_max_err attr_addfilterget_clip (
    void * x,
    double min,
    double max,
    long usemin,
    long usemax )
```

Attaches a clip filter to an attribute.

The filter will *only* clip values retrieved from the attribute using the attribute's get function.

#### Parameters

<i>x</i>	Pointer to the attribute to receive the filter
<i>min</i>	Minimum value for the clip filter
<i>max</i>	Maximum value for the clip filter
<i>usemin</i>	Sets this value to 0 if the minimum clip value should <i>not</i> be used. Otherwise, set the value to non-zero.
<i>usemax</i>	Sets this value to 0 if the maximum clip value should <i>not</i> be used. Otherwise, set the value to non-zero.

#### Returns

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

### 38.1.6.4 attr\_addfilterget\_clip\_scale()

```
t_max_err attr_addfilterget_clip_scale (
    void * x,
    double scale,
    double min,
    double max,
    long usemin,
    long usemax )
```

Attaches a clip/scale filter to an attribute.

The filter will *only* clip and scale values retrieved from the attribute using the attribute's get function.

#### Parameters

<i>x</i>	Pointer to the attribute to receive the filter
<i>scale</i>	Scale value. Data retrieved from the attribute will be scaled by this amount. <i>Scaling occurs previous to clipping.</i>
<i>min</i>	Minimum value for the clip filter
<i>max</i>	Maximum value for the clip filter
<i>usemin</i> <small>cycling 74</small>	Sets this value to 0 if the minimum clip value should <i>not</i> be used. Otherwise, set the value to non-zero.
<i>usemax</i>	Sets this value to 0 if the maximum clip value should <i>not</i> be used. Otherwise, set the value to non-zero.

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

**38.1.6.5 attr\_addfilterget\_proc()**

```
t_max_err attr_addfilterget_proc (
    void * x,
    method proc )
```

Attaches a custom filter method to an attribute.

The filter will *only* be called for values retrieved from the attribute using the attribute's `get` function.

**Parameters**

<code>x</code>	Pointer to the attribute to receive the filter
<code>proc</code>	A filter method

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

**Remarks**

The filter method should be prototyped and implemented as described above for the `attr_addfilterset_proc()` function.

**38.1.6.6 attr\_addfilterset\_clip()**

```
t_max_err attr_addfilterset_clip (
    void * x,
    double min,
    double max,
    long usemin,
    long usemax )
```

Attaches a clip filter to an attribute.

The filter will *only* clip values sent to the attribute using the attribute's `set` function.

**Parameters**

<i>x</i>	Pointer to the attribute to receive the filter
<i>min</i>	Minimum value for the clip filter
<i>max</i>	Maximum value for the clip filter
<i>usemin</i>	Sets this value to 0 if the minimum clip value should <i>not</i> be used. Otherwise, set the value to non-zero.
<i>usemax</i>	Sets this value to 0 if the maximum clip value should <i>not</i> be used. Otherwise, set the value to non-zero.

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

**38.1.6.7 attr\_addfilterset\_clip\_scale()**

```
t_max_err attr_addfilterset_clip_scale (
    void * x,
    double scale,
    double min,
    double max,
    long usemin,
    long usemax )
```

Attaches a clip/scale filter to an attribute.

The filter will *only* clip and scale values sent to the attribute using the attribute's `set` function.

**Parameters**

<i>x</i>	Pointer to the attribute to receive the filter
<i>scale</i>	Scale value. Data sent to the attribute will be scaled by this amount. <i>Scaling occurs previous to clipping.</i>
<i>min</i>	Minimum value for the clip filter
<i>max</i>	Maximum value for the clip filter
<i>usemin</i>	Sets this value to 0 if the minimum clip value should <i>not</i> be used. Otherwise, set the value to non-zero.
<i>usemax</i>	Sets this value to 0 if the maximum clip value should <i>not</i> be used. Otherwise, set the value to non-zero.

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

### 38.1.6.8 attr\_addfilterset\_proc()

```
t_max_err attr_addfilterset_proc (
    void * x,
    method proc )
```

Attaches a custom filter method to an attribute.

The filter will *only* be called for values retrieved from the attribute using the attribute's `set` function.

#### Parameters

<code>x</code>	Pointer to the attribute to receive the filter
<code>proc</code>	A filter method

#### Returns

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

#### Remarks

The filter method should be prototyped and implemented as follows:

```
t_max_err myfiltermethod(void *parent, void *attr, long ac, t_atom *av);
t_max_err myfiltermethod(void *parent, void *attr, long ac, t_atom *av)
{
    long i;
    float temp,
    // this filter rounds off all values
    // assumes that the data is float
    for (i = 0; i < ac; i++) {
        temp = atom_getfloat(av + i);
        temp = (float)((long)(temp + 0.5));
        atom_setfloat(av + i, temp);
    }
    return MAX_ERR_NONE;
}
```

### 38.1.6.9 attr\_args\_dictionary()

```
void attr_args_dictionary (
    t_dictionary * x,
    short ac,
    t_atom * av )
```

Create a dictionary of attribute-name, attribute-value pairs from an array of atoms containing an attribute definition list.

#### Parameters

<code>x</code>	A dictionary instance pointer.
<code>ac</code>	The number of atoms to parse in <code>av</code> .
<code>av</code>	A pointer to the first of the array of atoms containing the attribute values.

## Remarks

The code example below shows the creation of a list of atoms using [atom\\_setparse\(\)](#), and then uses that list of atoms to fill the dictionary with [attr\\_args\\_dictionary\(\)](#).

```
long ac = 0;
t_atom *av = NULL;
char parsebuf[4096];
t_dictionary *d = dictionary_new();
t_atom a;
sprintf(parsebuf, "@defrect %.6f %.6f %.6f %.6f @title Untitled @presentation 0 ", r->x, r->y, r->width,
       r->height);
atom_setparse(&ac, &av, parsebuf);
attr_args_dictionary(d, ac, av);
atom_setobj(&a, d);
```

### 38.1.6.10 attr\_args\_offset()

```
long attr_args_offset (
    short ac,
    t_atom * av )
```

Determines the point in an atom list where attribute arguments begin.

Developers can use this function to assist in the manual processing of attribute arguments, when [attr\\_args\\_process\(\)](#) doesn't provide the correct functionality for a particular purpose.

#### Parameters

<i>ac</i>	The count of <i>t_atoms</i> in <i>av</i>
<i>av</i>	An atom list

#### Returns

This function returns an offset into the atom list, where the first attribute argument occurs. For instance, the atom list `foo bar 3.0 @mode 6` would cause `attr_args_offset` to return 3 (the attribute `mode` appears at position 3 in the atom list).

Referenced by `max_jit_attr_args_offset()`.

### 38.1.6.11 attr\_args\_process()

```
void attr_args_process (
    void * x,
    short ac,
    t_atom * av )
```

Takes an atom list and properly set any attributes described within.

This function is typically used in an object's `new` method to conveniently process attribute arguments.

**Parameters**

<i>x</i>	The object whose attributes will be processed
<i>ac</i>	The count of <i>t_atoms</i> in <i>av</i>
<i>av</i>	An atom list

**Remarks**

Here is a typical example of usage:

```
void *myobject_new(t_symbol *s, long ac, t_atom *av)
{
    t_myobject *x = NULL;
    if (x=(t_myobject *)object_alloc(myobject_class))
    {
        // initialize any data before processing
        // attributes to avoid overwriting
        // attribute argument-set values
        x->data = 0;
        // process attr args, if any
        attr_args_process(x, ac, av);
    }
    return x;
}
```

**38.1.6.12 attr\_dictionary\_check()**

```
void attr_dictionary_check (
    void * x,
    t_dictionary * d )
```

Check that a dictionary only contains values for existing attributes of an object.

If a key in the dictionary doesn't correspond to one of the object's attributes, an error will be posted to the Max window.

**Parameters**

<i>x</i>	The object instance pointer.
<i>d</i>	The dictionary containing the attributes.

**See also**

[attr\\_dictionary\\_process\(\)](#)

**38.1.6.13 attr\_dictionary\_process()**

```
void attr_dictionary_process (
    void * x,
    t_dictionary * d )
```

Set attributes for an object that are defined in a dictionary.

Objects with dictionary constructors, such as UI objects, should call this method to set their attributes when an object is created.

#### Parameters

<i>x</i>	The object instance pointer.
<i>d</i>	The dictionary containing the attributes.

#### See also

[attr\\_args\\_process\(\)](#)

### 38.1.6.14 attr\_offset\_array\_new()

```
t_object* attr_offset_array_new (
    C74_CONST char * name,
    t_symbol * type,
    long size,
    long flags,
    method mget,
    method mset,
    long offsetcount,
    long offset )
```

Create a new attribute.

The attribute references an array of memory stored outside of itself, in the object's data structure. Attributes created using [attr\\_offset\\_array\\_new\(\)](#) can be assigned either to classes (using the [class\\_addattr\(\)](#) function) or to objects (using the [object\\_addattr\(\)](#) function).

#### Parameters

<i>name</i>	A name for the attribute, as a C-string
<i>type</i>	A <a href="#">t_symbol</a> * representing a valid attribute type. At the time of this writing, the valid type-symbols are: <a href="#">_sym_char</a> (char), <a href="#">_sym_long</a> (long), <a href="#">_sym_float32</a> (32-bit float), <a href="#">_sym_float64</a> (64-bit float), <a href="#">_sym_atom</a> (Max <a href="#">t_atom</a> pointer), <a href="#">_sym_symbol</a> (Max <a href="#">t_symbol</a> pointer), <a href="#">_sym_pointer</a> (generic pointer) and <a href="#">_sym_object</a> (Max <a href="#">t_object</a> pointer).
<i>size</i>	Maximum number of items that may be in the array.
<i>flags</i>	Any attribute flags, expressed as a bitfield. Attribute flags are used to determine if an attribute is accessible for setting or querying. The available accessor flags are defined in <a href="#">e_max_attrflags</a> .
<i>mget</i>	The method to use for the attribute's get functionality. If <i>mget</i> is NULL, the default method is used. See the discussion under <a href="#">attribute_new()</a> , for more information.
<i>mset</i>	The method to use for the attribute's set functionality. If <i>mset</i> is NULL, the default method is used. See the discussion under <a href="#">attribute_new()</a> , for more information.
<i>offsetcount</i>	Byte offset into the object class's data structure of a long variable describing how many array elements (up to <i>size</i> ) comprise the data to be referenced by the attribute. Typically, the <a href="#">calcoffset</a> macro is used to calculate this offset.
<i>Cycling '74 offset</i>	Byte offset into the class data structure of the object which will "own" the attribute. The offset should point to the data to be referenced by the attribute. Typically, the <a href="#">calcoffset</a> macro is used to calculate this offset.

**Returns**

This function returns the new attribute's object pointer if successful, or NULL if unsuccessful.

**Remarks**

For instance, to create a new attribute which references an array of 10 `t_atoms` (`atm`; the current number of "active" elements in the array is held in the variable `atmcount`) in an object class's data structure:

```
t_object *attr = attr_offset_array_new("myattrarray", _sym_atom /* matches data size */ , 10 /* max */ ,
0 /* no flags */ , (method)0L, (method)0L, calcoffset(t_myobject, atmcount) /* count */ ,
calcoffset(t_myobject, atm) /* data */ );
```

Referenced by `ext_main()`.

**38.1.6.15 attr\_offset\_new()**

```
t_object* attr_offset_new (
    C74_CONST char * name,
    C74_CONST t_symbol * type,
    long flags,
    C74_CONST method mget,
    C74_CONST method mset,
    long offset )
```

Create a new attribute.

The attribute references memory stored outside of itself, in the object's data structure. Attributes created using `attr_offset_new()` can be assigned either to classes (using the `class_addattr()` function) or to objects (using the `object_addattr()` function).

**Parameters**

<i>name</i>	A name for the attribute, as a C-string
<i>type</i>	A <code>t_symbol</code> * representing a valid attribute type. At the time of this writing, the valid type-symbols are: <code>_sym_char</code> ( <code>char</code> ), <code>_sym_long</code> ( <code>long</code> ), <code>_sym_float32</code> (32-bit float), <code>_sym_float64</code> (64-bit float), <code>_sym_atom</code> (Max <code>t_atom</code> pointer), <code>_sym_symbol</code> (Max <code>t_symbol</code> pointer), <code>_sym_pointer</code> (generic pointer) and <code>_sym_object</code> (Max <code>t_object</code> pointer).
<i>flags</i>	Any attribute flags, expressed as a bitfield. Attribute flags are used to determine if an attribute is accessible for setting or querying. The available accessor flags are defined in <code>e_max_attrflags</code> .
<i>mget</i>	The method to use for the attribute's get functionality. If <i>mget</i> is NULL, the default method is used. See the discussion under <code>attribute_new()</code> , for more information.
<i>mset</i>	The method to use for the attribute's set functionality. If <i>mset</i> is NULL, the default method is used. See the discussion under <code>attribute_new()</code> , for more information.
<i>offset</i>	Byte offset into the class data structure of the object which will "own" the attribute. The offset should point to the data to be referenced by the attribute. Typically, the <code>calcoffset</code> macro (described above) is used to calculate this offset.

**Returns**

This function returns the new attribute's object pointer if successful, or NULL if unsuccessful.

**Remarks**

For instance, to create a new attribute which references the value of a double variable (`val`) in an object class's data structure:

```
t_object *attr = attr_offset_new("myattr", _sym_float64 /* matches data size */, 0 /* no flags */,
                                (method)0L, (method)0L, calcoffset(t_myobject, val));
```

Referenced by `ext_main()`.

**38.1.6.16 attribute\_new()**

```
t_object* attribute_new (
    C74_CONST char * name,
    t_symbol * type,
    long flags,
    method mget,
    method mset )
```

Create a new attribute.

The attribute will allocate memory and store its own data. Attributes created using `attribute_new()` can be assigned either to classes (using the `class_addattr()` function) or to objects (using the `object_addattr()` function).

**Parameters**

<i>name</i>	A name for the attribute, as a C-string
<i>type</i>	A <code>t_symbol</code> * representing a valid attribute type. At the time of this writing, the valid type-symbols are: <code>_sym_char</code> (char), <code>_sym_long</code> (long), <code>_sym_float32</code> (32-bit float), <code>_sym_float64</code> (64-bit float), <code>_sym_atom</code> (Max <code>t_atom</code> pointer), <code>_sym_symbol</code> (Max <code>t_symbol</code> pointer), <code>_sym_pointer</code> (generic pointer) and <code>_sym_object</code> (Max <code>t_object</code> pointer).
<i>flags</i>	Any attribute flags, expressed as a bitfield. Attribute flags are used to determine if an attribute is accessible for setting or querying. The available accessor flags are defined in <code>e_max_attrflags</code> .
<i>mget</i>	The method to use for the attribute's get functionality. If <i>mget</i> is NULL, the default method is used.
<i>mset</i>	The method to use for the attribute's set functionality. If <i>mset</i> is NULL, the default method is used.

**Returns**

This function returns the new attribute's object pointer if successful, or NULL if unsuccessful.

## Remarks

Developers wishing to define custom methods for get or set functionality need to prototype them as:

```
t_max_err myobject_myattr_get(t_myobject *x, void *attr, long *ac, t_atom **av);
t_max_err myobject_myattr_set(t_myobject *x, void *attr, long ac, t_atom *av);
```

Implementation will vary, of course, but need to follow the following basic models. Note that, as with custom `getvalueof` and `setvalueof` methods for the object, assumptions are made throughout Max that `getbytes()` has been used for memory allocation. Developers are strongly urged to do the same:

```
t_max_err myobject_myattr_get(t_myobject *x, void *attr, long *ac, t_atom **av)
{
    if (*ac && *av)
        // memory passed in; use it
    else {
        *ac = 1; // size of attr data
        *av = (t_atom *)getbytes(sizeof(t_atom) * (*ac));
        if (!(*av)) {
            *ac = 0;
            return MAX_ERR_OUT_OF_MEM;
        }
    }
    atom_setlong(*av, x->some_value);
    return MAX_ERR_NONE;
}
t_max_err myobject_myattr_set(t_myobject *x, void *attr, long ac, t_atom *av)
{
    if (ac && av) {
        x->some_value = atom_getlong(av);
    }
    return MAX_ERR_NONE;
}
```

### 38.1.6.17 `object_addattr()`

```
t_max_err object_addattr (
    void * x,
    t_object * attr )
```

Attaches an attribute directly to an object.

#### Parameters

<code>x</code>	An object to which the attribute should be attached
<code>attr</code>	The attribute's pointer—this should be a pointer returned from <code>attribute_new()</code> , <code>attr_offset_new()</code> or <code>attr_offset_array_new()</code> .

#### Returns

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

### 38.1.6.18 `object_attr_get()`

```
void* object_attr_get (
    void * x,
    t_symbol * attrname )
```

Returns the pointer to an attribute, given its name.

#### Parameters

<i>x</i>	Pointer to the object whose attribute is of interest
<i>attrname</i>	The attribute's name

#### Returns

This function returns a pointer to the attribute, if successful, or NULL, if unsuccessful.

Referenced by jit\_attr\_getchar\_array(), jit\_attr\_getdouble\_array(), jit\_attr\_getfloat(), jit\_attr\_getfloat\_array(), jit\_attr\_getlong(), jit\_attr\_getlong\_array(), jit\_attr\_getsym(), jit\_attr\_getsym\_array(), jit\_attr\_setchar\_array(), jit\_attr\_setdouble\_array(), jit\_attr\_setfloat(), jit\_attr\_setfloat\_array(), jit\_attr\_setlong(), jit\_attr\_setlong\_array(), jit\_attr\_setsym(), jit\_attr\_setsym\_array(), jit\_object\_attr\_get(), jit\_object\_exportattrs(), and max\_jit\_attr\_set().

### 38.1.6.19 object\_attr\_get\_rect()

```
t_max_err object_attr_get_rect (
    t_object * o,
    t_symbol * name,
    t_rect * rect )
```

Gets the value of a `t_rect` attribute, given its parent object and name.

Do not use this on a jbox object – use `jbox_get_rect_for_view()` instead!

#### Parameters

<i>o</i>	The attribute's parent object
<i>name</i>	The attribute's name
<i>rect</i>	The address of a valid <code>t_rect</code> whose values will be filled-in from the attribute.

#### Returns

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

### 38.1.6.20 object\_attr\_getchar\_array()

```
long object_attr_getchar_array (
    void * x,
```

```
t_symbol * s,
long max,
t_uint8 * vals )
```

Retrieves the value of an attribute, given its parent object and name.

This function uses a developer-allocated array to copy data to. Developers wishing to retrieve the value of an attribute without pre-allocating memory should refer to the `object_attr_getvalueof()` function.

#### Parameters

<i>x</i>	The attribute's parent object
<i>s</i>	The attribute's name
<i>max</i>	The number of array elements in <code>vals</code> . The function will take care not to overwrite the bounds of the array.
<i>vals</i>	Pointer to the first element of a pre-allocated array of unsigned char data.

#### Returns

This function returns the number of elements copied into `vals`.

#### Remarks

If the attribute is not of the type specified by the function, the function will attempt to coerce a valid value from the attribute.

Referenced by `jit_attr_getchar_array()`.

### 38.1.6.21 `object_attr_getcolor()`

```
t_max_err object_attr_getcolor (
    t_object * b,
    t_symbol * attrname,
    t_jrgba * prgba )
```

Gets the value of a `t_jrgba` attribute, given its parent object and name.

#### Parameters

<i>b</i>	The attribute's parent object
<i>attrname</i>	The attribute's name
<i>prgba</i>	The address of a valid <code>t_jrgba</code> whose values will be filled-in from the attribute.

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

**38.1.6.22 object\_attr\_getdouble\_array()**

```
long object_attr_getdouble_array (
    void * x,
    t_symbol * s,
    long max,
    double * vals )
```

Retrieves the value of an attribute, given its parent object and name.

This function uses a developer-allocated array to copy data to. Developers wishing to retrieve the value of an attribute without pre-allocating memory should refer to the `object_attr_getvalueof()` function.

**Parameters**

<code>x</code>	The attribute's parent object
<code>s</code>	The attribute's name
<code>max</code>	The number of array elements in <code>vals</code> . The function will take care not to overwrite the bounds of the array.
<code>vals</code>	Pointer to the first element of a pre-allocated array of double data.

**Returns**

This function returns the number of elements copied into `vals`.

**Remarks**

If the attribute is not of the type specified by the function, the function will attempt to coerce a valid value from the attribute.

Referenced by `jit_attr_getdouble_array()`.

**38.1.6.23 object\_attr\_getdump()**

```
void object_attr_getdump (
    void * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Forces a specified object's attribute to send its value from the object's dumpout outlet in the Max interface.

**Parameters**

<i>x</i>	Pointer to the object whose attribute is of interest
<i>s</i>	The attribute's name
<i>argc</i>	Unused
<i>argv</i>	Unused

**38.1.6.24 object\_attr\_getfloat()**

```
t_atom_float object_attr_getfloat (
    void * x,
    t_symbol * s )
```

Retrieves the value of an attribute, given its parent object and name.

**Parameters**

<i>x</i>	The attribute's parent object
<i>s</i>	The attribute's name

**Returns**

This function returns the value of the specified attribute, if successful, or 0, if unsuccessful.

**Remarks**

If the attribute is not of the type specified by the function, the function will attempt to coerce a valid value from the attribute.

Referenced by jit\_attr\_getfloat().

**38.1.6.25 object\_attr\_getfloat\_array()**

```
long object_attr_getfloat_array (
    void * x,
    t_symbol * s,
    long max,
    float * vals )
```

Retrieves the value of an attribute, given its parent object and name.

This function uses a developer-allocated array to copy data to. Developers wishing to retrieve the value of an attribute without pre-allocating memory should refer to the object\_attr\_getvalueof() function.

**Parameters**

<i>x</i>	The attribute's parent object
<i>s</i>	The attribute's name
<i>max</i>	The number of array elements in <i>vals</i> . The function will take care not to overwrite the bounds of the array.
<i>vals</i>	Pointer to the first element of a pre-allocated array of float data.

**Returns**

This function returns the number of elements copied into *vals*.

**Remarks**

If the attribute is not of the type specified by the function, the function will attempt to coerce a valid value from the attribute.

Referenced by `jit_attr_getfloat_array()`.

**38.1.6.26 object\_attr\_getjrgba()**

```
t_max_err object_attr_getjrgba (
    void * ob,
    t_symbol * s,
    t_jrgba * c )
```

Retrieves the value of a color attribute, given its parent object and name.

**Parameters**

<i>ob</i>	The attribute's parent object
<i>s</i>	The attribute's name
<i>c</i>	The address of a <code>t_jrgba</code> struct that will be filled with the attribute's color component values.

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

**38.1.6.27 object\_attr\_getlong()**

```
t_atom_long object_attr_getlong (
    void * x,
    t_symbol * s )
```

Retrieves the value of an attribute, given its parent object and name.

#### Parameters

<i>x</i>	The attribute's parent object
<i>s</i>	The attribute's name

#### Returns

This function returns the value of the specified attribute, if successful, or 0, if unsuccessful.

#### Remarks

If the attribute is not of the type specified by the function, the function will attempt to coerce a valid value from the attribute.

Referenced by `jit_attr_getlong()`.

### 38.1.6.28 `object_attr_getlong_array()`

```
long object_attr_getlong_array (
    void * x,
    t_symbol * s,
    long max,
    t_atom_long * vals )
```

Retrieves the value of an attribute, given its parent object and name.

This function uses a developer-allocated array to copy data to. Developers wishing to retrieve the value of an attribute without pre-allocating memory should refer to the `object_attr_getvalueof()` function.

#### Parameters

<i>x</i>	The attribute's parent object
<i>s</i>	The attribute's name
<i>max</i>	The number of array elements in <i>vals</i> . The function will take care not to overwrite the bounds of the array.
<i>vals</i>	Pointer to the first element of a pre-allocated array of long data.

#### Returns

This function returns the number of elements copied into *vals*.

### Remarks

If the attribute is not of the type specified by the function, the function will attempt to coerce a valid value from the attribute.

Referenced by `jit_attr_getlong_array()`.

#### 38.1.6.29 `object_attr_getpt()`

```
t_max_err object_attr_getpt (
    t_object * o,
    t_symbol * name,
    t_pt * pt )
```

Gets the value of a `t_pt` attribute, given its parent object and name.

### Parameters

<code>o</code>	The attribute's parent object
<code>name</code>	The attribute's name
<code>pt</code>	The address of a valid <code>t_pt</code> whose values will be filled-in from the attribute.

### Returns

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

#### 38.1.6.30 `object_attr_getsize()`

```
t_max_err object_attr_getsize (
    t_object * o,
    t_symbol * name,
    t_size * size )
```

Gets the value of a `t_size` attribute, given its parent object and name.

### Parameters

<code>o</code>	The attribute's parent object
<code>name</code>	The attribute's name
<code>size</code>	The address of a valid <code>t_size</code> whose values will be filled-in from the attribute.

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

**38.1.6.31 object\_attr\_getsym()**

```
t_symbol* object_attr_getsym (
    void * x,
    t_symbol * s )
```

Retrieves the value of an attribute, given its parent object and name.

**Parameters**

<code>x</code>	The attribute's parent object
<code>s</code>	The attribute's name

**Returns**

This function returns the value of the specified attribute, if successful, or the empty symbol (equivalent to `gensym("")` or `_sym_nothing`), if unsuccessful.

Referenced by `jit_attr_getsym()`.

**38.1.6.32 object\_attr\_getsym\_array()**

```
long object_attr_getsym_array (
    void * x,
    t_symbol * s,
    long max,
    t_symbol ** vals )
```

Retrieves the value of an attribute, given its parent object and name.

This function uses a developer-allocated array to copy data to. Developers wishing to retrieve the value of an attribute without pre-allocating memory should refer to the `object_attr_getvalueof()` function.

**Parameters**

<code>x</code>	The attribute's parent object
<code>s</code>	The attribute's name
<code>max</code>	The number of array elements in <code>vals</code> . The function will take care not to overwrite the bounds of the array.
<code>vals</code>	Pointer to the first element of a pre-allocated array of <code>t_symbol *</code> s.

**Returns**

This function returns the number of elements copied into `vals`.

Referenced by `jit_attr_getsym_array()`.

**38.1.6.33 object\_attr\_method()**

```
method object_attr_method (
    void * x,
    t_symbol * methodname,
    void ** attr,
    long * get )
```

Returns the method of an attribute's get or set function, as well as a pointer to the attribute itself, from a message name.

**Parameters**

<code>x</code>	Pointer to the object whose attribute is of interest
<code>methodname</code>	The Max message used to call the attribute's get or set function. For example, <code>gensym ("mode")</code> or <code>gensym ("getthresh")</code> .
<code>attr</code>	A pointer to a void *, which will be set to the attribute pointer upon successful completion of the function
<code>get</code>	A pointer to a long variable, which will be set to 1 upon successful completion of the function, if the queried method corresponds to the <code>get</code> function of the attribute.

**Returns**

This function returns the requested method, if successful, or NULL, if unsuccessful.

**38.1.6.34 object\_attr\_set\_rect()**

```
t_max_err object_attr_set_rect (
    t_object * o,
    t_symbol * name,
    t_rect * rect )
```

Sets the value of a `t_rect` attribute, given its parent object and name.

Do not use this on a `jbox` object – use `jbox_get_rect_for_view()` instead!

**Parameters**

<i>o</i>	The attribute's parent object
<i>name</i>	The attribute's name
<i>rect</i>	The address of a valid <a href="#">t_rect</a> whose values will be used to set the attribute.

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

**38.1.6.35 object\_attr\_set\_xywh()**

```
void object_attr_set_xywh (
    t_object * o,
    t_symbol * attr,
    double x,
    double y,
    double w,
    double h )
```

Sets the value of a [t\\_rect](#) attribute, given its parent object and name.

Do not use this on a [jbox](#) object – use [jbox\\_get\\_rect\\_for\\_view\(\)](#) instead!

**Parameters**

<i>o</i>	The attribute's parent object
<i>attr</i>	The attribute's name
<i>x</i>	A double containing the new x position.
<i>y</i>	A double containing the new y position.
<i>w</i>	A double containing the new width.
<i>h</i>	A double containing the new height.

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

**38.1.6.36 object\_attr\_setchar\_array()**

```
t_max_err object_attr_setchar_array (
    void * x,
```

```
t_symbol * s,
long count,
C74_CONST t_uint8 * vals )
```

Sets the value of an attribute, given its parent object and name.

The function will call the attribute's `set` method, using the data provided.

#### Parameters

<i>x</i>	The attribute's parent object
<i>s</i>	The attribute's name
<i>count</i>	The number of array elements in <i>vals</i>
<i>vals</i>	Pointer to the first element of an array of unsigned char data

#### Returns

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `jit_attr_setchar_array()`.

### 38.1.6.37 `object_attr_setcolor()`

```
t_max_err object_attr_setcolor (
    t_object * b,
    t_symbol * attrname,
    t_jrgba * prgba )
```

Sets the value of a `t_jrgba` attribute, given its parent object and name.

#### Parameters

<i>b</i>	The attribute's parent object
<i>attrname</i>	The attribute's name
<i>prgba</i>	The address of a valid <code>t_jrgba</code> whose values will be used to set the attribute.

#### Returns

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

### 38.1.6.38 object\_attr\_setdouble\_array()

```
t_max_err object_attr_setdouble_array (
    void * x,
    t_symbol * s,
    long count,
    double * vals )
```

Sets the value of an attribute, given its parent object and name.

The function will call the attribute's `set` method, using the data provided.

#### Parameters

<i>x</i>	The attribute's parent object
<i>s</i>	The attribute's name
<i>count</i>	The number of array elements in <i>vals</i>
<i>vals</i>	Pointer to the first element of an array of double data

#### Returns

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `jit_attr_setdouble_array()`.

### 38.1.6.39 object\_attr\_setfloat()

```
t_max_err object_attr_setfloat (
    void * x,
    t_symbol * s,
    t_atom_float c )
```

Sets the value of an attribute, given its parent object and name.

The function will call the attribute's `set` method, using the data provided.

#### Parameters

<i>x</i>	The attribute's parent object
<i>s</i>	The attribute's name
<i>c</i>	An floating point value; the new value for the attribute

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `jit_attr_setfloat()`.

**38.1.6.40 object\_attr\_setfloat\_array()**

```
t_max_err object_attr_setfloat_array (
    void * x,
    t_symbol * s,
    long count,
    float * vals )
```

Sets the value of an attribute, given its parent object and name.

The function will call the attribute's `set` method, using the data provided.

**Parameters**

<code>x</code>	The attribute's parent object
<code>s</code>	The attribute's name
<code>count</code>	The number of array elements in <code>vals</code>
<code>vals</code>	Pointer to the first element of an array of float data

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `jit_attr_setfloat_array()`.

**38.1.6.41 object\_attr\_setjrgba()**

```
t_max_err object_attr_setjrgba (
    void * ob,
    t_symbol * s,
    t_jrgba * c )
```

Sets the value of a color attribute, given its parent object and name.

The function will call the attribute's `set` method, using the data provided.

**Parameters**

<i>ob</i>	The attribute's parent object
<i>s</i>	The attribute's name
<i>c</i>	The address of a <a href="#">t_jrgba</a> struct that contains the new color.

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

**38.1.6.42 object\_attr\_setlong()**

```
t_max_err object_attr_setlong (
    void * x,
    t_symbol * s,
    t_atom_long c )
```

Sets the value of an attribute, given its parent object and name.

The function will call the attribute's `set` method, using the data provided.

**Parameters**

<i>x</i>	The attribute's parent object
<i>s</i>	The attribute's name
<i>c</i>	An integer value; the new value for the attribute

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

Referenced by `jit_attr_setlong()`.

**38.1.6.43 object\_attr\_setlong\_array()**

```
t_max_err object_attr_setlong_array (
    void * x,
    t_symbol * s,
    long count,
    t_atom_long * vals )
```

Sets the value of an attribute, given its parent object and name.

The function will call the attribute's `set` method, using the data provided.

**Parameters**

<i>x</i>	The attribute's parent object
<i>s</i>	The attribute's name
<i>count</i>	The number of array elements in <i>vals</i>
<i>vals</i>	Pointer to the first element of an array of long data

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

Referenced by [jit\\_attr\\_setlong\\_array\(\)](#).

**38.1.6.44 object\_attr\_setparse()**

```
t_max_err object_attr_setparse (
    t_object * x,
    t_symbol * s,
    C74_CONST char * parsestr )
```

Set an attribute value with one or more atoms parsed from a C-string.

**Parameters**

<i>x</i>	The object whose attribute will be set.
<i>s</i>	The name of the attribute to set.
<i>parsestr</i>	A C-string to parse into an array of atoms to set the attribute value.

**Returns**

A Max error code.

**See also**

[atom\\_setparse\(\)](#)

**38.1.6.45 object\_attr\_setpt()**

```
t_max_err object_attr_setpt (
    t_object * o,
    t_symbol * name,
    t_pt * pt )
```

Sets the value of a [t\\_pt](#) attribute, given its parent object and name.

**Parameters**

<i>o</i>	The attribute's parent object
<i>name</i>	The attribute's name
<i>pt</i>	The address of a valid <a href="#">t_pt</a> whose values will be used to set the attribute.

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

**38.1.6.46 object\_attr\_setsize()**

```
t_max_err object_attr_setsize (
    t_object * o,
    t_symbol * name,
    t_size * size )
```

Sets the value of a [t\\_size](#) attribute, given its parent object and name.

**Parameters**

<i>o</i>	The attribute's parent object
<i>name</i>	The attribute's name
<i>size</i>	The address of a valid <a href="#">t_size</a> whose values will be used to set the attribute.

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

**38.1.6.47 object\_attr\_setsym()**

```
t_max_err object_attr_setsym (
    void * x,
    t_symbol * s,
    t_symbol * c )
```

Sets the value of an attribute, given its parent object and name.

The function will call the attribute's `set` method, using the data provided.

**Parameters**

<i>x</i>	The attribute's parent object
<i>s</i>	The attribute's name
<i>c</i>	A <a href="#">t_symbol</a> *; the new value for the attribute

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

Referenced by [jit\\_attr\\_setsym\(\)](#).

**38.1.6.48 object\_attr\_setsym\_array()**

```
t_max_err object_attr_setsym_array (
    void * x,
    t_symbol * s,
    long count,
    t_symbol ** vals )
```

Sets the value of an attribute, given its parent object and name.

The function will call the attribute's `set` method, using the data provided.

**Parameters**

<i>x</i>	The attribute's parent object
<i>s</i>	The attribute's name
<i>count</i>	The number of array elements in <i>vals</i>
<i>vals</i>	Pointer to the first element of an array of <a href="#">t_symbol</a> *s

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

Referenced by [jit\\_attr\\_setsym\\_array\(\)](#).

**38.1.6.49 object\_attr\_setvalueof()**

```
t_max_err object_attr_setvalueof (
    void * x,
```

```
t_symbol * s,
long argc,
t_atom * argv )
```

Sets the value of an object's attribute.

#### Parameters

<i>x</i>	Pointer to the object whose attribute is of interest
<i>s</i>	The attribute's name
<i>argc</i>	The count of arguments in <i>argv</i>
<i>argv</i>	Array of <i>t_atoms</i> ; the new desired data for the attribute

#### Returns

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

Referenced by [jit\\_object\\_importattrs\(\)](#).

### 38.1.6.50 `object_attr_usercanget()`

```
long object_attr_usercanget (
    void * x,
    t_symbol * s )
```

Determines if the value of an object's attribute can be queried from the Max interface (i.e.

if its [ATTR\\_GET\\_OPAQUE\\_USER](#) flag is set).

#### Parameters

<i>x</i>	Pointer to the object whose attribute is of interest
<i>s</i>	The attribute's name

#### Returns

This function returns 1 if the value of the attribute can be queried from the Max interface. Otherwise, it returns 0.

Referenced by [jit\\_object\\_attr\\_usercanget\(\)](#).

### 38.1.6.51 object\_attr\_usercanset()

```
long object_attr_usercanset (
    void * x,
    t_symbol * s )
```

Determines if an object's attribute can be set from the Max interface (i.e.

if its [ATTR\\_SET\\_OPAQUE\\_USER](#) flag is set).

#### Parameters

<i>x</i>	Pointer to the object whose attribute is of interest
<i>s</i>	The attribute's name

#### Returns

This function returns 1 if the attribute can be set from the Max interface. Otherwise, it returns 0.

Referenced by [jit\\_object\\_attr\\_usercanset\(\)](#).

### 38.1.6.52 object\_chuckattr()

```
t_max_err object_chuckattr (
    void * x,
    t_symbol * attrsym )
```

Detach an attribute from an object that was previously attached with [object\\_addattr\(\)](#).

This function will *not* free the attribute (use [object\\_free\(\)](#) to do this manually).

#### Parameters

<i>x</i>	The object to which the attribute is attached
<i>attrsym</i>	The attribute's name

#### Returns

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

### 38.1.6.53 `object_deleteattr()`

```
t_max_err object_deleteattr (
    void * x,
    t_symbol * attrsym )
```

Detach an attribute from an object that was previously attached with `object_addattr()`.

The function will also free all memory associated with the attribute. If you only wish to detach the attribute, without freeing it, see the `object_chuckattr()` function.

#### Parameters

<code>x</code>	The object to which the attribute is attached
<code>attrsym</code>	The attribute's name

#### Returns

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

### 38.1.6.54 `object_new_parse()`

```
void* object_new_parse (
    t_symbol * name_space,
    t_symbol * classname,
    C74_CONST char * parsestr )
```

Create a new object with one or more atoms parsed from a C-string.

The object's new method must have an `A_GIMME` signature.

#### Parameters

<code>name_space</code>	The namespace in which to create the instance. Typically this is either <code>CLASS_BOX</code> or <code>CLASS_NOBOX</code> .
<code>classname</code>	The name of the class to instantiate.
<code>parsestr</code>	A C-string to parse into an array of atoms to set the attribute value.

#### Returns

A pointer to the new instance.

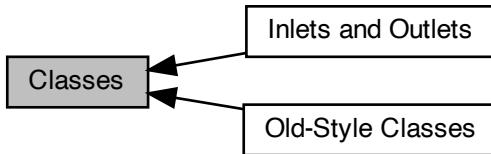
#### See also

`atom_setparse()`  
`object_new_typed()`

## 38.2 Classes

When a user types the name of your object into an object box, Max looks for an external of this name in the searchpath and, upon finding it, loads the bundle or dll and calls the `ext_main()` function.

Collaboration diagram for Classes:



## Modules

- [Old-Style Classes](#)
- [Inlets and Outlets](#)

*Routines for creating and communicating with inlets and outlets.*

## Data Structures

- [struct t\\_class](#)

*The data structure for a Max class.*

## Macros

- [#define CLASS\\_BOX](#)

*The namespace for all Max object classes which can be instantiated in a box, i.e.*

- [#define CLASS\\_NOBOX](#)

*A namespace for creating hidden or internal object classes which are not a direct part of the user creating patcher.*

## Enumerations

- [enum e\\_max\\_class\\_flags {](#)
- [CLASS\\_FLAG\\_BOX](#) , [CLASS\\_FLAG\\_POLYGLOT](#) , [CLASS\\_FLAG\\_NEWDICTIONARY](#) , [CLASS\\_FLAG\\_REGISTERED](#) ,  
[CLASS\\_FLAG\\_UIOBJECT](#) , [CLASS\\_FLAG\\_ALIAS](#) , [CLASS\\_FLAG\\_MULTITOUCH](#) , [CLASS\\_FLAG\\_DO\\_NOT\\_PARSE\\_ATTR\\_ARC](#) ,  
[CLASS\\_FLAG\\_DO\\_NOT\\_ZERO](#) , [CLASS\\_FLAG\\_NOATTRIBUTES](#) , [CLASS\\_FLAG\\_OWNATTRIBUTES](#) ,  
[CLASS\\_FLAG\\_PARAMETER](#) ,  
[CLASS\\_FLAG\\_RETYPEABLE](#) , [CLASS\\_FLAG\\_OBJECT\\_METHOD](#) , [CLASS\\_FLAG\\_VISUALIZER](#) , [CLASS\\_FLAGUSES\\_PROXY](#) ,  
[CLASS\\_FLAG\\_OWN\\_DATA](#) , [CLASS\\_FLAG\\_DYNAMICCOLOR](#) }

*Class flags.*

## Functions

- `BEGIN_USING_C_LINKAGE void C74_EXPORT ext_main (void *r)`  
`ext_main()` is the entry point for an extern to be loaded, which all externs must implement this shared/common prototype ensures that it will be exported correctly on all platforms.
- `t_class * class_new (C74_CONST char *name, C74_CONST method mnew, C74_CONST method mfree, long size, C74_CONST method mmenu, short type,...)`  
*Initializes a class by informing Max of its name, instance creation and free functions, size and argument types.*
- `t_max_err class_free (t_class *c)`  
*Frees a previously defined object class.*
- `t_max_err class_register (t_symbol *name_space, t_class *c)`  
*Registers a previously defined object class.*
- `t_max_err class_alias (t_class *c, t_symbol *aliasname)`  
*Registers an alias for a previously defined object class.*
- `t_max_err class_addmethod (t_class *c, C74_CONST method m, C74_CONST char *name,...)`  
*Adds a method to a previously defined object class.*
- `t_max_err class_addattr (t_class *c, t_object *attr)`  
*Adds an attribute to a previously defined object class.*
- `t_symbol * class_nameget (t_class *c)`  
*Retrieves the name of a class, given the class's pointer.*
- `t_class * class_findbyname (t_symbol *name_space, t_symbol *classname)`  
*Finds the class pointer for a class, given the class's namespace and name.*
- `t_class * class_findbyname_casemode (t_symbol *name_space, t_symbol *classname)`  
*Finds the class pointer for a class, given the class's namespace and name.*
- `t_max_err class_dumpout_wrap (t_class *c)`  
*Wraps user gettable attributes with a method that gets the values and sends out dumpout outlet.*
- `void class_obexoffset_set (t_class *c, long offset)`  
*Registers the byte-offset of the obex member of the class's data structure with the previously defined object class.*
- `long class_obexoffset_get (t_class *c)`  
*Retrieves the byte-offset of the obex member of the class's data structure.*
- `long class_is_ui (t_class *c)`  
*Determine if a class is a user interface object.*
- `t_max_err class_subclass (t_class *superclass, t_class *subclass)`  
*Define a subclass of an existing class.*
- `t_object * class_super_construct (t_class *c,...)`  
*Call super class constructor.*

### 38.2.1 Detailed Description

When a user types the name of your object into an object box, Max looks for an external of this name in the searchpath and, upon finding it, loads the bundle or dll and calls the `ext_main()` function.

Thus, Max classes are typically defined in the `ext_main()` function of an external.

Historically, Max classes have been defined using an API that includes functions like `setup()` and `addmess()`. This interface is still supported, and the relevant documentation can be found in [Old-Style Classes](#).

A more recent and more flexible interface for creating objects was introduced with Jitter 1.0 and later included directly in Max 4.5. This newer API includes functions such as `class_new()` and `class_addmethod()`. Supporting attributes, user interface objects, and additional new features of Max requires the use of the newer interface for defining classes documented on this page.

You may not mix these two styles of creating classes within an object.

### 38.2.2 Macro Definition Documentation

#### 38.2.2.1 CLASS\_BOX

```
#define CLASS_BOX
```

The namespace for all Max object classes which can be instantiated in a box, i.e.

in a patcher.

### 38.2.3 Enumeration Type Documentation

#### 38.2.3.1 e\_max\_class\_flags

```
enum e_max_class_flags
```

Class flags.

If not box or polyglot, class is only accessible in C via known interface

Enumerator

CLASS_FLAG_BOX	for use in a patcher
CLASS_FLAG_POLYGLOT	for use by any text language (c/js/java/etc)
CLASS_FLAG_NEWDICTIONARY	dictionary based constructor
CLASS_FLAG_REGISTERED	for backward compatible messlist implementation (once reg'd can't grow)
CLASS_FLAG_UIOBJECT	for objects that don't go inside a newobj box.
CLASS_FLAG_ALIAS	for classes that are just copies of some other class (i.e. del is a copy of delay)
CLASS_FLAG_MULTITOUCHE	sent multitouch version of mouse messages
CLASS_FLAG_DO_NOT_PARSE_ATTR_ARGS	override dictionary based constructor attr arg parsing
CLASS_FLAG_DO_NOT_ZERO	don't zero the object struct on construction (for efficiency)
CLASS_FLAG_NOATTRIBUTES	for efficiency
CLASS_FLAG_OWNATTRIBUTES	for classes which support a custom attr interface (e.g. jitter)
CLASS_FLAG_PARAMETER	for classes which have a parameter
CLASS_FLAG_RETYPEABLE	object box can be retyped without recreating the object
CLASS_FLAG_OBJECT_METHOD	objects of this class may have object specific methods
CLASS_FLAG_VISUALIZER	objects of this class are signal visualizers
CLASS_FLAGUSES_PROXYIES	objects of this class might use proxies (set automatically in proxy_new)
CLASS_FLAG_OWN_DATA	objects of this class save data in their own format
CLASS_FLAG_DYNAMICCOLOR	objects which contain colors supporting dynamic colors (set automatically in class_attr_dynamiccolor_init)

### 38.2.4 Function Documentation

#### 38.2.4.1 class\_addattr()

```
t_max_err class_addattr (
    t_class * c,
    t_object * attr )
```

Adds an attribute to a previously defined object class.

##### Parameters

<i>c</i>	The class pointer
<i>attr</i>	The attribute to add. The attribute will be a pointer returned by <a href="#">attribute_new()</a> , <a href="#">attr_offset_new()</a> or <a href="#">attr_offset_array_new()</a> .

##### Returns

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

Referenced by `ext_main()`, and `jit_class_addattr()`.

#### 38.2.4.2 class\_addmethod()

```
t_max_err class_addmethod (
    t_class * c,
    C74_CONST method m,
    C74_CONST char * name,
    ... )
```

Adds a method to a previously defined object class.

##### Parameters

<i>c</i>	The class pointer
<i>m</i>	Function to be called when the method is invoked
<i>name</i>	C-string defining the message (message selector)
...	One or more integers specifying the arguments to the message, in the standard Max type list format (see Chapter 3 of the Writing Externals in Max document for more information).

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

**Remarks**

The `class_addmethod()` function works essentially like the traditional `addmess()` function, adding the function pointed to by `m`, to respond to the message string `name` in the leftmost inlet of the object.

Referenced by `ext_main()`, `jit_class_addmethod()`, and `jit_class_new()`.

**38.2.4.3 class\_alias()**

```
t_max_err class_alias (
    t_class * c,
    t_symbol * aliasname )
```

Registers an alias for a previously defined object class.

**Parameters**

<code>c</code>	The class pointer
<code>aliasname</code>	A symbol who's name will become an alias for the given class

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

**38.2.4.4 class\_dumpout\_wrap()**

```
t_max_err class_dumpout_wrap (
    t_class * c )
```

Wraps user gettable attributes with a method that gets the values and sends out dumpout outlet.

**Parameters**

<code>c</code>	The class pointer
----------------	-------------------

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

**38.2.4.5 class\_findbyname()**

```
t_class* class_findbyname (
    t_symbol * name_space,
    t_symbol * classname )
```

Finds the class pointer for a class, given the class's namespace and name.

**Parameters**

<i>name_space</i>	The desired class's name space. Typically, either the constant <a href="#">CLASS_BOX</a> , for obex classes which can instantiate inside of a Max patcher (e.g. boxes, UI objects, etc.), or the constant <a href="#">CLASS_NOBOX</a> , for classes which will only be used internally. Developers can define their own name spaces as well, but this functionality is currently undocumented.
<i>classname</i>	The name of the class to be looked up

**Returns**

If successful, this function returns the class's data pointer. Otherwise, it returns NULL.

Referenced by [jit\\_class\\_findbyname\(\)](#).

**38.2.4.6 class\_findbyname\_casefree()**

```
t_class* class_findbyname_casefree (
    t_symbol * name_space,
    t_symbol * classname )
```

Finds the class pointer for a class, given the class's namespace and name.

**Parameters**

<i>name_space</i>	The desired class's name space. Typically, either the constant <a href="#">CLASS_BOX</a> , for obex classes which can instantiate inside of a Max patcher (e.g. boxes, UI objects, etc.), or the constant <a href="#">CLASS_NOBOX</a> , for classes which will only be used internally. Developers can define their own name spaces as well, but this functionality is currently undocumented.
<i>classname</i>	The name of the class to be looked up (case free)

**Returns**

If successful, this function returns the class's data pointer. Otherwise, it returns NULL.

#### 38.2.4.7 `class_free()`

```
t_max_err class_free (
    t_class * c )
```

Frees a previously defined object class.

*This function is not typically used by external developers.*

**Parameters**

<code>c</code>	The class pointer
----------------	-------------------

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `jit_class_free()`.

#### 38.2.4.8 `class_is_ui()`

```
long class_is_ui (
    t_class * c )
```

Determine if a class is a user interface object.

**Parameters**

<code>c</code>	The class pointer.
----------------	--------------------

**Returns**

True is the class defines a user interface object, otherwise false.

### 38.2.4.9 class\_nameget()

```
t_symbol* class_nameget (
    t_class * c )
```

Retrieves the name of a class, given the class's pointer.

#### Parameters

<code>c</code>	The class pointer
----------------	-------------------

#### Returns

If successful, this function returns the name of the class as a `t_symbol` \*.

Referenced by `jit_class_nameget()`.

### 38.2.4.10 class\_new()

```
t_class* class_new (
    C74_CONST char * name,
    C74_CONST method mnew,
    C74_CONST method mfree,
    long size,
    C74_CONST method mmenu,
    short type,
    ... )
```

Initializes a class by informing Max of its name, instance creation and free functions, size and argument types.

Developers wishing to use obex class features (attributes, etc.) *must* use `class_new()` instead of the traditional `setup()` function.

#### Parameters

<code>name</code>	The class's name, as a C-string
<code>mnew</code>	The instance creation function
<code>mfree</code>	The instance free function
<code>size</code>	The size of the object's data structure in bytes. Usually you use the C sizeof operator here.
<code>mmenu</code>	Obsolete - pass NULL. In Max 4 this was a function pointer for UI objects called when the user created a new object of the class from the Patch window's palette.
<code>type</code>	A standard Max <i>type list</i> as explained in Chapter 3 of the Writing Externals in Max document (in the Max SDK). The final argument of the type list should be a 0. <i>Generally, obex objects have a single type argument, A_GIMME, followed by a 0.</i>

**Returns**

This function returns the class pointer for the new object class. *This pointer is used by numerous other functions and should be stored in a global or static variable.*

Referenced by `ext_main()`, and `jit_class_new()`.

**38.2.4.11 `class_obexoffset_get()`**

```
long class_obexoffset_get (
    t_class * c )
```

Retrieves the byte-offset of the obex member of the class's data structure.

**Parameters**

<code>c</code>	The class pointer
----------------	-------------------

**Returns**

This function returns the byte-offset of the obex member of the class's data structure.

**38.2.4.12 `class_obexoffset_set()`**

```
void class_obexoffset_set (
    t_class * c,
    long offset )
```

Registers the byte-offset of the obex member of the class's data structure with the previously defined object class.

Use of this function is required for obex-class objects. It must be called from `main()`.

**Parameters**

<code>c</code>	The class pointer
<code>offset</code>	The byte-offset to the obex member of the object's data structure. Conventionally, the macro <code>calcoffset</code> is used to calculate the offset.

Referenced by `ext_main()`.

### 38.2.4.13 class\_register()

```
t_max_err class_register (
    t_symbol * name_space,
    t_class * c )
```

Registers a previously defined object class.

This function is required, and should be called at the end of `main()`.

#### Parameters

<i>name_space</i>	The desired class's name space. Typically, either the constant <code>CLASS_BOX</code> , for obex classes which can instantiate inside of a Max patcher (e.g. boxes, UI objects, etc.), or the constant <code>CLASS_NOBOX</code> , for classes which will only be used internally. Developers can define their own name spaces as well, but this functionality is currently undocumented.
<i>c</i>	The class pointer

#### Returns

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `ext_main()`, and `jit_class_register()`.

### 38.2.4.14 class\_subclass()

```
t_max_err class_subclass (
    t_class * superclass,
    t_class * subclass )
```

Define a subclass of an existing class.

First call `class_new` on the subclass, then pass it to `class_subclass`. If constructor or destructor are NULL will use the superclass constructor.

#### Parameters

<i>superclass</i>	The superclass pointer.
<i>subclass</i>	The subclass pointer.

#### Returns

A Max error code

### 38.2.4.15 class\_super\_construct()

```
t_object* class_super_construct (
    t_class * c,
    ... )
```

Call super class constructor.

Use this instead of object\_alloc if you want to call the super class constructor, but allocating enough memory for sub-class.

#### Parameters

<i>c</i>	The (sub)class pointer.
...	Args to super class constructor.

#### Returns

initialized object instance

### 38.2.4.16 ext\_main()

```
BEGIN_USING_C_LINKAGE void C74_EXPORT ext_main (
    void * r )
```

[ext\\_main\(\)](#) is the entry point for an extern to be loaded, which all externs must implement this shared/common prototype ensures that it will be exported correctly on all platforms.

#### Parameters

<i>r</i>	Pointer to resources for the external, if applicable.
----------	---

#### See also

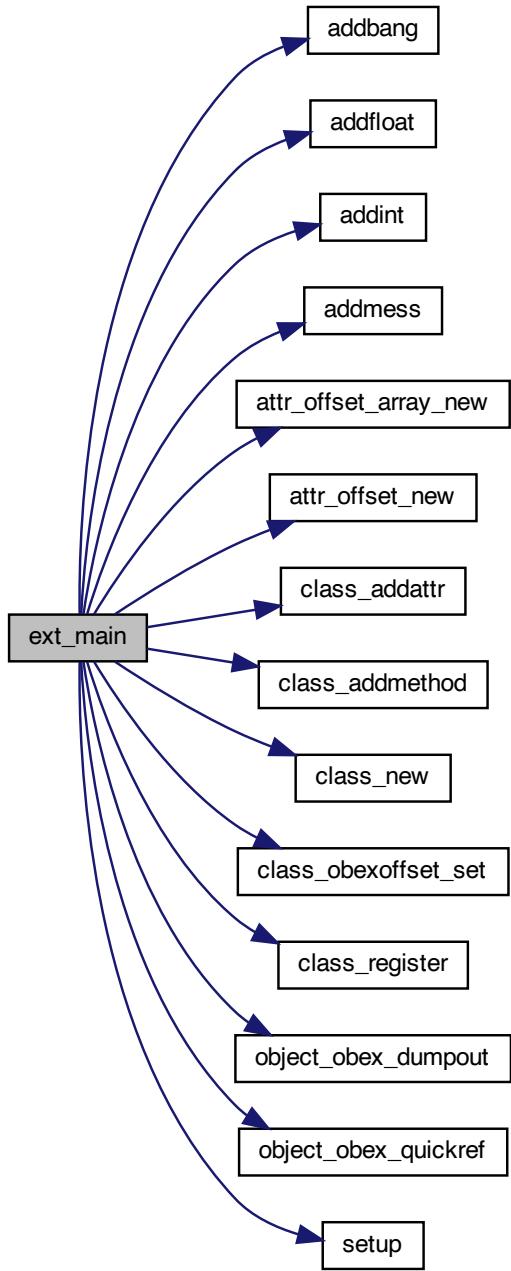
[Anatomy of a Max Object](#)

#### Version

Introduced in Max 6.1.9

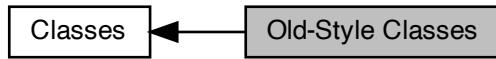
References A\_CANT, A\_FLOAT, A\_GIMME, A\_LONG, addbang(), addfloat(), addint(), addmess(), attr\_offset\_array<new(), attr\_offset\_new(), ATTR\_SET\_OPAQUE, ATTR\_SET\_OPAQUE\_USER, calcoffset, class\_addattr(), class<addmethod(), CLASS\_BOX, class\_new(), class\_obexoffset\_set(), class\_register(), object\_obex\_dumpout(), object<obex\_quickref(), and setup().

Here is the call graph for this function:



## 38.3 Old-Style Classes

Collaboration diagram for Old-Style Classes:



### Functions

- **BEGIN\_USING\_C\_LINKAGE void `setup` (`t_messlist **ident, method makefun, method freefun, t_getbytes_size size, method menufun, short type,...)`**

*Use the `setup()` function to initialize your class by informing Max of its size, the name of your functions that create and destroy instances, and the types of arguments passed to the instance creation function.*

- **void `address` (method f, char \*s, short type,...)**

*Use `address()` to bind a function to a message other than the standard ones covered by `addbang()`, `addint()`, etc.*

- **void `addbang` (method f)**

*Used to bind a function to the common triggering message bang.*

- **void `addint` (method f)**

*Use `addint()` to bind a function to the int message received in the leftmost inlet.*

- **void `addfloat` (method f)**

*Use `addfloat()` to bind a function to the float message received in the leftmost inlet.*

- **void `addinx` (method f, short n)**

*Use `addinx()` to bind a function to a int message that will be received in an inlet other than the leftmost one.*

- **void `additx` (method f, short n)**

*Use `additx()` to bind a function to a float message that will be received in an inlet other than the leftmost one.*

- **void \* `newobject` (void \*maxclass)**

*Use `newobject` to allocate the space for an instance of your class and initialize its object header.*

- **void `freeobject` (void \*op)**

*Release the memory used by a Max object.*

- **void \* `newinstance` (`t_symbol *s, short argc, t_atom *argv`)**

*Make a new instance of an existing Max class.*

- **void `alias` (char \*name)**

*Use the `alias` function to allow users to refer to your object by a name other than that of your shared library.*

- **void `class_setname` (char \*obname, char \*filename)**

*Use `class_setname()` to associate your object's name with its filename on disk.*

- **void \* `typedmess` (`t_object *op, t_symbol *msg, short argc, t_atom *argv`)**

*Send a typed message directly to a Max object.*

- **method `getfn` (`t_object *op, t_symbol *msg`)**

*Use `getfn()` to send an untyped message to a Max object with error checking.*

- **method `egetfn` (`t_object *op, t_symbol *msg`)**

*Use `egetfn()` to send an untyped message to a Max object that always works.*

- **method `zgetfn` (`t_object *op, t_symbol *msg`)**

*Use `zgetfn()` to send an untyped message to a Max object without error checking.*

### 38.3.1 Detailed Description

### 38.3.2 Function Documentation

#### 38.3.2.1 addbang()

```
void addbang (
    method f )
```

Used to bind a function to the common triggering message bang.

##### Parameters

<i>f</i>	Function to be the bang method.
----------	---------------------------------

Referenced by ext\_main(), and max\_jit\_classex\_mop\_wrap().

#### 38.3.2.2 addfloat()

```
void addfloat (
    method f )
```

Use [addfloat\(\)](#) to bind a function to the float message received in the leftmost inlet.

##### Parameters

<i>f</i>	Function to be the int method.
----------	--------------------------------

Referenced by ext\_main().

#### 38.3.2.3 addftx()

```
void addftx (
    method f,
    short n )
```

Use [addftx\(\)](#) to bind a function to a float message that will be received in an inlet other than the leftmost one.

**Parameters**

<i>f</i>	Function to be the float method.
<i>n</i>	Number of the inlet connected to this method. 1 is the first inlet to the right of the left inlet.

**Remarks**

This correspondence between inlet locations and messages is not automatic, but it is strongly suggested that you follow existing practice. You must set the correspondence up when creating an object of your class with proper use of intin and floatin in your instance creation function [New Instance Routine](#).

**38.3.2.4 addint()**

```
void addint (
    method f )
```

Use [addint\(\)](#) to bind a function to the int message received in the leftmost inlet.

**Parameters**

<i>f</i>	Function to be the int method.
----------	--------------------------------

Referenced by ext\_main().

**38.3.2.5 addinx()**

```
void addinx (
    method f,
    short n )
```

Use [addinx\(\)](#) to bind a function to a int message that will be received in an inlet other than the leftmost one.

**Parameters**

<i>f</i>	Function to be the int method.
<i>n</i>	Number of the inlet connected to this method. 1 is the first inlet to the right of the left inlet.

**Remarks**

This correspondence between inlet locations and messages is not automatic, but it is strongly suggested that you follow existing practice. You must set the correspondence up when creating an object of your class with proper use of intin and floatin in your instance creation function [New Instance Routine](#).

### 38.3.2.6 addmess()

```
void addmess (
    method f,
    char * s,
    short type,
    ... )
```

Use [addmess\(\)](#) to bind a function to a message other than the standard ones covered by [addbang\(\)](#), [addint\(\)](#), etc.

#### Parameters

<i>f</i>	Function you want to be the method.
<i>s</i>	C string defining the message.
<i>type</i>	The first of one or more integers from <a href="#">e_max_atomtypes</a> specifying the arguments to the message.
...	Any additional types from <a href="#">e_max_atomtypes</a> for additional arguments.

#### See also

[Anatomy of a Max Object](#)

Referenced by [ext\\_main\(\)](#), [max\\_addmethod\\_defer\(\)](#), [max\\_addmethod\\_defer\\_low\(\)](#), [max\\_addmethod\\_usurp\(\)](#), [max\\_addmethod\\_usurp\\_low\(\)](#), [max\\_jit\\_classex\\_addattr\(\)](#), [max\\_jit\\_classex\\_mop\\_wrap\(\)](#), [max\\_jit\\_classex\\_setup\(\)](#), and [max\\_jit\\_classex\\_standard\\_wrap\(\)](#).

### 38.3.2.7 alias()

```
void alias (
    char * name )
```

Use the alias function to allow users to refer to your object by a name other than that of your shared library.

#### Parameters

<i>name</i>	An alternative name for the user to use to make an object of your class.
-------------	--

### 38.3.2.8 class\_setname()

```
void class_setname (
```

```
char * obname,
char * filename )
```

Use [class\\_setname\(\)](#) to associate your object's name with its filename on disk.

#### Parameters

<i>obname</i>	A character string with the name of your object class as it appears in Max.
<i>filename</i>	A character string with the name of your external's file as it appears on disk.

### 38.3.2.9 `egetfn()`

```
method egetfn (
    t_object * op,
    t_symbol * msg )
```

Use [egetfn\(\)](#) to send an untyped message to a Max object that always works.

#### Parameters

<i>op</i>	Receiver of the message.
<i>msg</i>	Message selector.

#### Returns

`egetfn` returns a pointer to the method bound to the message selector `msg` in the receiver's message list. If the method can't be found, a pointer to a do-nothing function is returned.

### 38.3.2.10 `freeobject()`

```
void freeobject (
    void * op )
```

Release the memory used by a Max object.

[freeobject\(\)](#) calls an object's free function, if any, then disposes the memory used by the object itself. [freeobject\(\)](#) should be used on any instance of a standard Max object data structure, with the exception of Qelems and Atombufs. Clocks, Binbufs, Proxies, Exprs, etc. should be freed with [freeobject\(\)](#).

#### Parameters

<i>op</i>	The object instance pointer to free.
-----------	--------------------------------------

**Remarks**

This function can be replaced by the use of [object\\_free\(\)](#). Unlike [freeobject\(\)](#), [object\\_free\(\)](#) checkes to make sure the pointer is not NULL before trying to free it.

**See also**

[newobject\(\)](#)  
[object\\_free\(\)](#)

Referenced by `max_jit_obex_gimmeback()`, and `max_jit_obex_gimmeback_dumpout()`.

**38.3.2.11 getfn()**

```
method getfn (
    t_object * op,
    t_symbol * msg )
```

Use [getfn\(\)](#) to send an untyped message to a Max object with error checking.

**Parameters**

<i>op</i>	Receiver of the message.
<i>msg</i>	Message selector.

**Returns**

`getfn` returns a pointer to the method bound to the message selector `msg` in the receiver's message list. It returns 0 and prints an error message in Max Window if the method can't be found.

**38.3.2.12 newinstance()**

```
void* newinstance (
    t_symbol * s,
    short argc,
    t_atom * argv )
```

Make a new instance of an existing Max class.

**Parameters**

<i>s</i>	className Symbol specifying the name of the class of the instance to be created.
<i>argc</i>	Count of arguments in <code>argv</code> .
<i>argv</i>	Array of <code>t_atoms</code> ; arguments to the class's instance creation function.

**Returns**

A pointer to the created object, or 0 if the class didn't exist or there was another type of error in creating the instance.

**Remarks**

This function creates a new instance of the specified class. Using newinstance is equivalent to typing something in a New Object box when using Max. The difference is that no object box is created in any Patcher window, and you can send messages to the object directly without connecting any patch cords. The messages can either be type- checked (using typedmess) or non-type-checked (using the members of the getfn family).

This function is useful for taking advantage of other already-defined objects that you would like to use 'privately' in your object, such as tables. See the source code for the coll object for an example of using a privately defined class.

**38.3.2.13 newobject()**

```
void* newobject (
    void * maxclass )
```

Use newobject to allocate the space for an instance of your class and initialize its object header.

**Parameters**

<i>maxclass</i>	The global class variable initialized in your main routine by the setup function.
-----------------	---

**Returns**

A pointer to the new instance.

**Remarks**

You call [newobject\(\)](#) when creating an instance of your class in your creation function. newobject allocates the proper amount of memory for an object of your class and installs a pointer to your class in the object, so that it can respond with your class's methods if it receives a message.

Referenced by `max_jit_obex_new()`.

**38.3.2.14 setup()**

```
BEGIN_USING_C_LINKAGE void setup (
    t_messlist ** ident,
    method makefun,
    method freefun,
    t_getbytes_size size,
```

```
method menufun,  
short type,  
... )
```

Use the [setup\(\)](#) function to initialize your class by informing Max of its size, the name of your functions that create and destroy instances, and the types of arguments passed to the instance creation function.

**Parameters**

<i>ident</i>	A global variable in your code that points to the initialized class.
<i>makefun</i>	Your instance creation function.
<i>freefun</i>	Your instance free function (see Chapter 7).
<i>size</i>	The size of your objects data structure in bytes. Usually you use the C sizeof operator here.
<i>menufun</i>	No longer used. You should pass NULL for this parameter.
<i>type</i>	The first of a list of arguments passed to makefun when an object is created.
...	Any additional arguments passed to makefun when an object is created. Together with the type parameter, this creates a standard Max type list as enumerated in <a href="#">e_max_atomtypes</a> . The final argument of the type list should be a 0.

**See also**

[Anatomy of a Max Object](#)

Referenced by `ext_main()`.

**38.3.2.15 typedmess()**

```
void* typedmess (
    t_object * op,
    t_symbol * msg,
    short argc,
    t_atom * argp )
```

Send a typed message directly to a Max object.

**Parameters**

<i>op</i>	Max object that will receive the message.
<i>msg</i>	The message selector.
<i>argc</i>	Count of message arguments in argv.
<i>argp</i>	Array of t_atoms; the message arguments.

**Returns**

If the receiver object can respond to the message, [typedmess\(\)](#) returns the result. Otherwise, an error message will be seen in the Max window and 0 will be returned.

### Remarks

`typedmess` sends a message to a Max object (receiver) a message with arguments. Note that the message must be a `t_symbol`, not a character string, so you must call `gensym` on a string before passing it to `typedmess`. Also, note that untyped messages defined for classes with the argument list `A_CANT` cannot be sent using `typedmess`. You must use `getfn()` etc. instead.

### Example:

```
//If you want to send a bang message to the object bang_me...
void *bangResult;
bangResult = typedmess(bang_me, gensym("bang"), 0, 0L);
```

Referenced by `max_jit_mop_bang()`.

### 38.3.2.16 zgetfn()

```
method zgetfn (
    t_object * op,
    t_symbol * msg )
```

Use `zgetfn()` to send an untyped message to a Max object without error checking.

### Parameters

<code>op</code>	Receiver of the message.
<code>msg</code>	Message selector.

### Returns

`zgetfn` returns a pointer to the method bound to the message selector `msg` in the receiver's message list. It returns 0 but doesn't print an error message in Max Window if the method can't be found.

Referenced by `max_jit_attr_getdump()`.

## 38.4 Inlets and Outlets

Routines for creating and communicating with inlets and outlets.

Collaboration diagram for Inlets and Outlets:



## Functions

- void \* [inlet\\_new](#) (void \*x, C74\_CONST char \*s)
 

*Use [inlet\\_new\(\)](#) to create an inlet that can receive a specific message or any message.*
- void \* [intin](#) (void \*x, short n)
 

*Use [intin\(\)](#) to create an inlet typed to receive only integers.*
- void \* [floatin](#) (void \*x, short n)
 

*Use [floatin\(\)](#) to create an inlet typed to receive only floats.*
- void \* [outlet\\_new](#) (void \*x, C74\_CONST char \*s)
 

*Use [outlet\\_new\(\)](#) to create an outlet that can send a specific non-standard message, or any message.*
- void \* [bangout](#) (void \*x)
 

*Use [bangout\(\)](#) to create an outlet that will always send the bang message.*
- void \* [intout](#) (void \*x)
 

*Use [intout\(\)](#) to create an outlet that will always send the int message.*
- void \* [floatout](#) (void \*x)
 

*Use [floatout\(\)](#) to create an outlet that will always send the float message.*
- void \* [listout](#) (void \*x)
 

*Use [listout\(\)](#) to create an outlet that will always send the list message.*
- void \* [outlet\\_bang](#) (t\_outlet \*x)
 

*Use [outlet\\_bang\(\)](#) to send a bang message out an outlet.*
- void \* [outlet\\_int](#) (t\_outlet \*x, t\_atom\_long n)
 

*Use [outlet\\_int\(\)](#) to send an int message out an outlet.*
- void \* [outlet\\_float](#) (t\_outlet \*x, double f)
 

*Use [outlet\\_float\(\)](#) to send a float message out an outlet.*
- void \* [outlet\\_list](#) (t\_outlet \*x, t\_symbol \*s, short ac, t\_atom \*av)
 

*Use [outlet\\_list\(\)](#) to send a list message out an outlet.*
- void \* [outlet\\_anything](#) (t\_outlet \*x, t\_symbol \*s, short ac, t\_atom \*av)
 

*Use [outlet\\_anything\(\)](#) to send any message out an outlet.*
- void \* [proxy\\_new](#) (void \*x, long id, long \*stuffloc)
 

*Use [proxy\\_new](#) to create a new Proxy object.*
- long [proxy\\_getinlet](#) (t\_object \*master)
 

*Use [proxy\\_getinlet](#) to get the inlet number in which a message was received.*

### 38.4.1 Detailed Description

Routines for creating and communicating with inlets and outlets.

### 38.4.2 Function Documentation

#### 38.4.2.1 [bangout\(\)](#)

```
void* bangout (
    void * x )
```

Use [bangout\(\)](#) to create an outlet that will always send the bang message.

**Parameters**

x	Your object.
---	--------------

**Returns**

A pointer to the new outlet.

**Remarks**

You can send a bang message out a general purpose outlet, but creating an outlet using [bangout\(\)](#) allows Max to type-check the connection a user might make and refuse to connect the outlet to any object that cannot receive a bang message. [bangout\(\)](#) returns the created outlet.

**38.4.2.2 floatin()**

```
void* floatin (
    void * x,
    short n )
```

Use [floatin\(\)](#) to create an inlet typed to receive only floats.

**Parameters**

x	Your object.
n	Location of the inlet from 1 to 9. 1 is immediately to the right of the leftmost inlet.

**Returns**

A pointer to the new inlet.

**38.4.2.3 floatout()**

```
void* floatout (
    void * x )
```

Use [floatout\(\)](#) to create an outlet that will always send the float message.

**Parameters**

x	Your object.
---	--------------

**Returns**

A pointer to the new outlet.

**38.4.2.4 inlet\_new()**

```
void* inlet_new (
    void * x,
    C74_CONST char * s )
```

Use [inlet\\_new\(\)](#) to create an inlet that can receive a specific message or any message.

**Parameters**

x	Your object.
s	Character string of the message, or NULL to receive any message.

**Returns**

A pointer to the new inlet.

**Remarks**

[inlet\\_new\(\)](#) creates a general purpose inlet. You can use it in circumstances where you would like special messages to be received in inlets other than the leftmost one. To create an inlet that receives a particular message, pass the message's character string. For example, to create an inlet that receives only bang messages, do the following  
`inlet_new (myObject, "bang");`

To create an inlet that can receive any message, pass NULL for msg  
`inlet_new (myObject, NULL);`

Proxies are an alternative method for general-purpose inlets that have a number of advantages. If you create multiple inlets as shown above, there would be no way to figure out which inlet received a message. See the discussion in [Creating and Using Proxies](#).

**38.4.2.5 intin()**

```
void* intin (
    void * x,
    short n )
```

Use [intin\(\)](#) to create an inlet typed to receive only integers.

**Parameters**

<i>x</i>	Your object.
<i>n</i>	Location of the inlet from 1 to 9. 1 is immediately to the right of the leftmost inlet.

**Returns**

A pointer to the new inlet.

**Remarks**

`intin` creates integer inlets. It takes a pointer to your newly created object and an integer *n*, from 1 to 9. The number specifies the message type you'll get, so you can distinguish one inlet from another. For example, an integer sent in inlet 1 will be of message type `in1` and a floating point number sent in inlet 4 will be of type `ft4`. You use [addinx\(\)](#) and [addftx\(\)](#) to add methods to respond to these messages.

The order you create additional inlets is important. If you want the rightmost inlet to be the have the highest number in- or ft- message (which is usually the case), you should create the highest number message inlet first.

**38.4.2.6 `intout()`**

```
void* intout (
    void * x )
```

Use [intout\(\)](#) to create an outlet that will always send the `int` message.

**Parameters**

<i>x</i>	Your object.
----------	--------------

**Returns**

A pointer to the new outlet.

**Remarks**

You can send a bang message out a general purpose outlet, but creating an outlet using [bangout\(\)](#) allows Max to type-check the connection a user might make and refuse to connect the outlet to any object that cannot receive a bang message. [bangout\(\)](#) returns the created outlet.

**38.4.2.7 `listout()`**

```
void* listout (
    void * x )
```

Use [listout\(\)](#) to create an outlet that will always send the `list` message.

**Parameters**

x	Your object.
---	--------------

**Returns**

A pointer to the new outlet.

**38.4.2.8 outlet\_anything()**

```
void* outlet_anything (
    t_outlet * x,
    t_symbol * s,
    short ac,
    t_atom * av )
```

Use [outlet\\_anything\(\)](#) to send any message out an outlet.

**Parameters**

<i>o</i>	Outlet that will send the message.
<i>s</i>	The message selector <a href="#">t_symbol*</a> .
<i>ac</i>	Number of elements in the list in argv.
<i>av</i>	Atoms constituting the list.

**Returns**

Returns 0 if a stack overflow occurred, otherwise returns 1.

**Remarks**

This function lets you send an arbitrary message out an outlet. Here are a couple of examples of its use.

First, here's a hard way to send the bang message (see [outlet\\_bang\(\)](#) for an easier way):

```
outlet_anything(myOutlet, gensym("bang"), 0, NIL);
```

**Remarks**

And here's an even harder way to send a single integer (instead of using [outlet\\_int\(\)](#)):

```
t_atom myNumber;
atom_setlong(&myNumber, 432);
outlet_anything(myOutlet, gensym("int"), 1, &myNumber);
```

Notice that [outlet\\_anything\(\)](#) expects the message argument as a [t\\_symbol\\*](#), so you must use [gensym\(\)](#) on a character string.

If you'll be sending the same message a lot, you might call [gensym\(\)](#) on the message string at initialization time and store the result in a global variable to save the (significant) overhead of calling [gensym\(\)](#) every time you want to send a message.

Also, do not send lists using [outlet\\_anything\(\)](#) with list as the selector argument. Use the [outlet\\_list\(\)](#) function instead.

Referenced by `max_jit_mop_jit_matrix()`, `max_jit_mop_outputmatrix()`, and `max_jit_obex_dumpout()`.

### 38.4.2.9 [outlet\\_bang\(\)](#)

```
void* outlet_bang (
    t_outlet * x )
```

Use [outlet\\_bang\(\)](#) to send a bang message out an outlet.

#### Parameters

<i>o</i>	Outlet that will send the message.
----------	------------------------------------

#### Returns

Returns 0 if a stack overflow occurred, otherwise returns 1.

### 38.4.2.10 [outlet\\_float\(\)](#)

```
void* outlet_float (
    t_outlet * x,
    double f )
```

Use [outlet\\_float\(\)](#) to send a float message out an outlet.

#### Parameters

<i>o</i>	Outlet that will send the message.
<i>f</i>	Float value to send.

#### Returns

Returns 0 if a stack overflow occurred, otherwise returns 1.

### 38.4.2.11 outlet\_int()

```
void* outlet_int (
    t_outlet * x,
    t_atom_long n )
```

Use [outlet\\_int\(\)](#) to send an int message out an outlet.

#### Parameters

<i>o</i>	Outlet that will send the message.
<i>n</i>	Integer value to send.

#### Returns

Returns 0 if a stack overflow occurred, otherwise returns 1.

### 38.4.2.12 outlet\_list()

```
void* outlet_list (
    t_outlet * x,
    t_symbol * s,
    short ac,
    t_atom * av )
```

Use [outlet\\_list\(\)](#) to send a list message out an outlet.

#### Parameters

<i>o</i>	Outlet that will send the message.
<i>s</i>	Should be NULL, but can be the _sym_list.
<i>ac</i>	Number of elements in the list in argv.
<i>av</i>	Atoms constituting the list.

#### Returns

Returns 0 if a stack overflow occurred, otherwise returns 1.

#### Remarks

[outlet\\_list\(\)](#) sends the list specified by argv and argc out the specified outlet. The outlet must have been created with listout or outlet\_new in your object creation function (see above). You create the list as an array of Atoms, but the first item in the list must be an integer or float.

Here's an example of sending a list of three numbers.

```
t_atom myList[3];
long theNumbers[3];
short i;
theNumbers[0] = 23;
theNumbers[1] = 12;
theNumbers[2] = 5;
for (i=0; i < 3; i++) {
    atom_setlong(myList+i, theNumbers[i]);
}
outlet_list(myOutlet, 0L, 3, &myList);
```

#### Remarks

It's not a good idea to pass large lists to `outlet_list` that are comprised of local (automatic) variables. If the list is small, as in the above example, there's no problem. If your object will regularly send lists, it might make sense to keep an array of `t_atoms` inside your object's data structure.

#### 38.4.2.13 `outlet_new()`

```
void* outlet_new (
    void * x,
    C74_CONST char * s )
```

Use `outlet_new()` to create an outlet that can send a specific non-standard message, or any message.

#### Parameters

<code>x</code>	Your object.
<code>s</code>	A C-string specifying the message that will be sent out this outlet, or NULL to indicate the outlet will be used to send various messages. The advantage of this kind of outlet's flexibility is balanced by the fact that Max must perform a message-lookup in real-time for every message sent through it, rather than when a patch is being constructed, as is true for other types of outlets. Patchers execute faster when outlets are typed, since the message lookup can be done before the program executes.

#### Returns

A pointer to the new outlet.

Referenced by `max_jit_mop_matrixout_new()`, and `max_jit_mop_setup_simple()`.

#### 38.4.2.14 `proxy_getinlet()`

```
long proxy_getinlet (
    t_object * master )
```

Use `proxy_getinlet` to get the inlet number in which a message was received.

Note that the `owner` argument should point to your external object's instance, not a proxy object.

**Parameters**

<i>master</i>	Your object.
---------------	--------------

**Returns**

The index number of the inlet that received the message.

Referenced by `max_jit_obex_inletnumber_get()`.

**38.4.2.15 proxy\_new()**

```
void* proxy_new (
    void * x,
    long id,
    long * stuffloc )
```

Use `proxy_new` to create a new Proxy object.

**Parameters**

<i>x</i>	Your object.
<i>id</i>	A non-zero number to be written into your object when a message is received in this particular Proxy. Normally, <i>id</i> will be the inlet number analogous to <i>in1</i> , <i>in2</i> etc.
<i>stuffloc</i>	A pointer to a location where the <i>id</i> value will be written.

**Returns**

A pointer to the new proxy inlet.

**Remarks**

This routine creates a new Proxy object (that includes an inlet). It allows you to identify messages based on an *id* value stored in the location specified by *stuffLoc*. You should store the pointer returned by `proxy_new()` because you'll need to free all Proxies in your object's free function using `object_free()`.

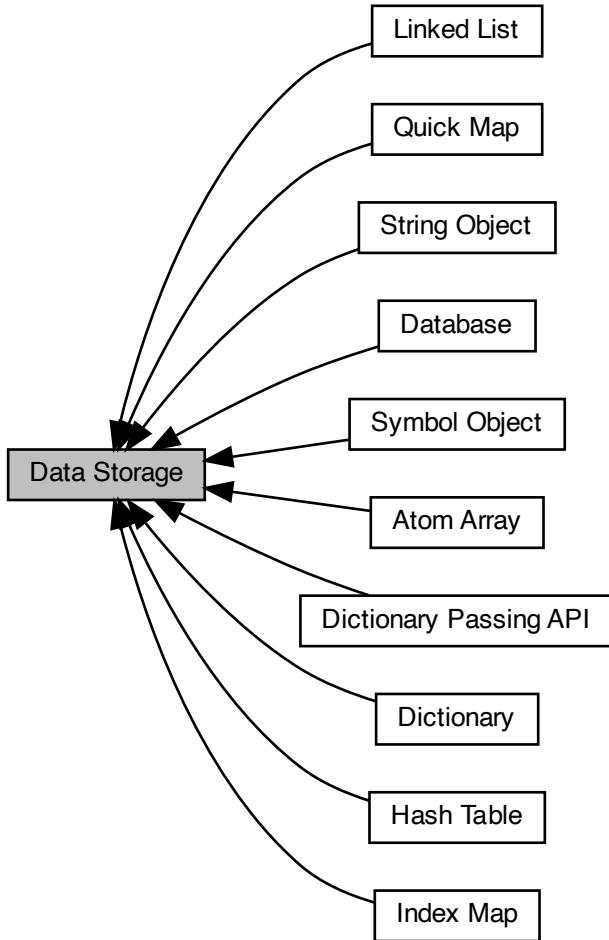
After your method has finished, Proxy sets the *stuffLoc* location back to 0, since it never sees messages coming in an object's leftmost inlet. You'll know you received a message in the leftmost inlet if the contents of *stuffLoc* is 0. As of Max 4.3, *stuffLoc* is not always guaranteed to be a correct indicator of the inlet in which a message was received. Use `proxy_getinlet()` to determine the inlet number.

Referenced by `max_jit_obex_proxy_new()`.

## 38.5 Data Storage

Max provides a number of ways of storing and manipulating data at a high level.

Collaboration diagram for Data Storage:



### Modules

- [Atom Array](#)

*Max's atomarray object is a container for an array of atoms with an interface for manipulating that array.*

- [Database](#)

*Max's database ( i.e.*

- [Dictionary](#)

Max 5, introduced the [t\\_dictionary](#) structure/object.

- [Hash Table](#)

*A hash table is a data structure that associates some data with a unique key.*

- [Index Map](#)

*An indexmap is basically a managed array of pointers, but it allows you to derive relatively quickly the index from a pointer in the array.*

- [Linked List](#)

*The Max [t\\_linklist](#) data structure is useful for maintaining ordered lists of items where you want to be able to insert and delete items efficiently.*

- [Quick Map](#)

*A quickmap implements a pair of [t\\_hashtab](#) hash tables so that it is fast to look up a unique value for a unique key or vice-versa.*

- [String Object](#)

*Max's string object is a simple wrapper for c-strings, useful when working with Max's [t\\_dictionary](#), [t\\_linklist](#), or [t\\_hashtab](#).*

- [Symbol Object](#)

*The symobject class is a simple object that wraps a [t\\_symbol](#)\* together with a couple of additional fields.*

- [Dictionary Passing API](#)

*The Dictionary Passing API defines a means by which [t\\_dictionary](#) instances may be passed between Max objects in a way similar to the way Jitter Matrices are passed between objects.*

## Typedefs

- [typedef long\(\\* t\\_cmpfn\) \(void \\*, void \\*\)](#)

*Comparison function pointer type.*

## Enumerations

- [enum e\\_max\\_datastore\\_flags {  
OBJ\\_FLAG\\_OBJ , OBJ\\_FLAG\\_REF , OBJ\\_FLAG\\_DATA , OBJ\\_FLAG\\_MEMORY ,  
OBJ\\_FLAG\\_SILENT , OBJ\\_FLAG\\_INHERITABLE , OBJ\\_FLAG\\_ITERATING , OBJ\\_FLAG\\_CLONE ,  
OBJ\\_FLAG\\_DANGER , OBJ\\_FLAG\\_DEBUG }](#)

*Flags used in linklist and hashtab objects.*

### 38.5.1 Detailed Description

Max provides a number of ways of storing and manipulating data at a high level.

It is recommended to use Max's data storage mechanisms where possible, as Max's systems are designed for thread-safety and integration with the rest of Max API.

### 38.5.2 Typedef Documentation

### 38.5.2.1 `t_cmpfn`

```
typedef long(* t_cmpfn) (void *, void *)
```

Comparison function pointer type.

Methods that require a comparison function pointer to be passed in use this type. It should return `true` or `false` depending on the outcome of the comparison of the two linklist items passed in as arguments.

See also

- [linklist\\_match\(\)](#)
- [hashtab\\_findfirst\(\)](#)
- [indexmap\\_sort\(\)](#)

## 38.5.3 Enumeration Type Documentation

### 38.5.3.1 `e_max_datastore_flags`

```
enum e_max_datastore_flags
```

Flags used in linklist and hashtab objects.

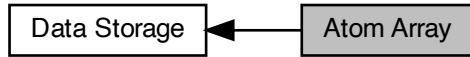
Enumerator

<code>OBJ_FLAG_OBJ</code>	free using <a href="#">object_free()</a>
<code>OBJ_FLAG_REF</code>	don't free
<code>OBJ_FLAG_DATA</code>	don't free data or call method
<code>OBJ_FLAG_MEMORY</code>	don't call method, and when freeing use <a href="#">sysmem_freeptr()</a> instead of freeobject
<code>OBJ_FLAG_SILENT</code>	don't notify when modified
<code>OBJ_FLAG_INHERITABLE</code>	obexprototype entry will be inherited by subpatchers and abstractions
<code>OBJ_FLAG_ITERATING</code>	used by linklist to signal when is inside iteration
<code>OBJ_FLAG_CLONE</code>	object should be cloned when added to data structure (i.e. dictionary)
<code>OBJ_FLAG_DANGER</code>	context-dependent flag, used internally for hashtable code
<code>OBJ_FLAG_DEBUG</code>	context-dependent flag, used internally for linklist debug code

## 38.6 Atom Array

Max's atomarray object is a container for an array of atoms with an interface for manipulating that array.

Collaboration diagram for Atom Array:



## Data Structures

- struct `t_atomarray`

*The atomarray object.*

## Enumerations

- enum `t_atomarray_flags`

*The atomarray flags.*

## Functions

- `BEGIN_USING_C_LINKAGE t_atomarray * atomarray_new (long ac, t_atom *av)`

*Create a new atomarray object.*
- `void atomarray_flags (t_atomarray *x, long flags)`

*Set the atomarray flags.*
- `long atomarray_getflags (t_atomarray *x)`

*Get the atomarray flags.*
- `t_max_err atomarray_setatoms (t_atomarray **x, long ac, t_atom *av)`

*Replace the existing array contents with a new set of atoms Note that atoms provided to this function will be copied.*
- `t_max_err atomarray_getatoms (t_atomarray **x, long *ac, t_atom **av)`

*Retrieve a pointer to the first atom in the internal array of atoms.*
- `t_max_err atomarray_copyatoms (t_atomarray **x, long *ac, t_atom **av)`

*Retrieve a copy of the atoms in the array.*
- `t_atom_long atomarray_getsize (t_atomarray **x)`

*Return the number of atoms in the array.*
- `t_max_err atomarray_getindex (t_atomarray **x, long index, t_atom *av)`

*Copy an a specific atom from the array.*
- `void * atomarray_duplicate (t_atomarray **x)`

*Create a new atomarray object which is a copy of another atomarray object.*
- `void * atomarray_clone (t_atomarray **x)`

*Create a new atomarray object which is a full clone of another atomarray object.*
- `void atomarray_appendatom (t_atomarray **x, t_atom *a)`

*Copy a new atom onto the end of the array.*

- void [atomarray\\_appendatoms](#) ([t\\_atomarray](#) \*x, long ac, [t\\_atom](#) \*av)  
*Copy multiple new atoms onto the end of the array.*
- void [atomarray\\_chuckindex](#) ([t\\_atomarray](#) \*x, long index)  
*Remove an atom from any location within the array.*
- void [atomarray\\_clear](#) ([t\\_atomarray](#) \*x)  
*Clear the array.*
- void [atomarray\\_funall](#) ([t\\_atomarray](#) \*x, method fun, void \*arg)  
*Call the specified function for every item in the atom array.*

### 38.6.1 Detailed Description

Max's atomarray object is a container for an array of atoms with an interface for manipulating that array.

It can be useful for passing lists as a single atom, such as for the return value of an [A\\_GIMMEBACK](#) method. It also used frequently in when working with Max's [t\\_dictionary](#) object.

See also

[Dictionary](#)

### 38.6.2 Enumeration Type Documentation

#### 38.6.2.1 [t\\_atomarray\\_flags](#)

```
enum t\_atomarray\_flags
```

The atomarray flags.

Currently the only flag is ATOMARRAY\_FLAG\_FREECHILDREN. If set via [atomarray\\_flags\(\)](#) the atomarray will free any contained A\_OBJ atoms when the atomarray is freed.

### 38.6.3 Function Documentation

#### 38.6.3.1 [atomarray\\_appendatom\(\)](#)

```
void atomarray_appendatom (
    t\_atomarray * x,
    t\_atom * a )
```

Copy a new atom onto the end of the array.

**Parameters**

<i>x</i>	The atomarray instance.
<i>a</i>	A pointer to the new atom to append to the end of the array.

**See also**

[atomarray\\_appendatoms\(\)](#)  
[atomarray\\_setatoms\(\)](#)

**38.6.3.2 atomarray\_appendatoms()**

```
void atomarray_appendatoms (
    t_atomarray * x,
    long ac,
    t_atom * av )
```

Copy multiple new atoms onto the end of the array.

**Parameters**

<i>x</i>	The atomarray instance.
<i>ac</i>	The number of new atoms to be appended to the array.
<i>av</i>	A pointer to the first of the new atoms to append to the end of the array.

**See also**

[atomarray\\_appendatom\(\)](#)  
[atomarray\\_setatoms\(\)](#)

**38.6.3.3 atomarray\_chuckindex()**

```
void atomarray_chuckindex (
    t_atomarray * x,
    long index )
```

Remove an atom from any location within the array.

The array will be resized and collapsed to fill in the gap.

**Parameters**

<i>x</i>	The atomarray instance.
<i>index</i>	The zero-based index of the atom to remove from the array.

**38.6.3.4 atomarray\_clear()**

```
void atomarray_clear (
    t_atomarray * x )
```

Clear the array.

Frees all of the atoms and sets the size to zero. This function does not perform a 'deep' free, meaning that any [A\\_OBJ](#) atoms will not have their object's freed. Only the references to those objects contained in the atomarray will be freed.

**Parameters**

<i>x</i>	The atomarray instance.
----------	-------------------------

**Returns**

The number of atoms in the array.

**38.6.3.5 atomarray\_clone()**

```
void* atomarray_clone (
    t_atomarray * x )
```

Create a new atomarray object which is a full clone of another atomarray object.

**Parameters**

<i>x</i>	The atomarray instance which is to be copied.
----------	---

**Returns**

A new atomarray which is copied from *x*.

**See also**

[atomarray\\_new\(\)](#)

### 38.6.3.6 atomarray\_copyatoms()

```
t_max_err atomarray_copyatoms (
    t_atomarray * x,
    long * ac,
    t_atom ** av )
```

Retrieve a copy of the atoms in the array.

To retrieve a pointer to the contained atoms use [atomarray\\_getatoms\(\)](#).

#### Parameters

<i>x</i>	The atomarray instance.
<i>ac</i>	The address of a long where the number of atoms will be set.
<i>av</i>	The address of a <a href="#">t_atom</a> pointer where the atoms will be allocated and copied.

#### Returns

A Max error code.

#### Remarks

You are responsible for freeing memory allocated for the copy of the atoms returned.

```
long ac = 0;
t_atom *av = NULL;
atomarray_copyatoms(anAtomarray, &ac, &av);
if(ac && av) {
    // do something with ac and av here...
    system_freeptr(av);
}
```

#### See also

[atomarray\\_getatoms\(\)](#)

### 38.6.3.7 atomarray\_duplicate()

```
void* atomarray_duplicate (
    t_atomarray * x )
```

Create a new atomarray object which is a copy of another atomarray object.

#### Parameters

<i>x</i>	The atomarray instance which is to be copied.
----------	---

**Returns**

A new atomarray which is copied from x.

**See also**

[atomarray\\_new\(\)](#)

**38.6.3.8 atomarray\_flags()**

```
void atomarray_flags (
    t_atomarray * x,
    long flags )
```

Set the atomarray flags.

**Parameters**

x	The atomarray instance.
flags	The new value for the flags.

**38.6.3.9 atomarray\_funall()**

```
void atomarray_funall (
    t_atomarray * x,
    method fun,
    void * arg )
```

Call the specified function for every item in the atom array.

**Parameters**

x	The atomarray instance.
fun	The function to call, specified as function pointer cast to a Max #method.
arg	An argument that you would like to pass to the function being called.

**Returns**

A max error code.

**Remarks**

The [atomarray\\_funall\(\)](#) method will call your function for every item in the list. It will pass both a pointer to the item in the list, and any argument that you provide. The following example shows a function that could be called by [hashtab\\_funall\(\)](#).

```
void myFun(t_atom *a, void *myArg)
{
    // do something with a and myArg here
    // a is the atom in the atom array
}
```

**See also**

[linklist\\_funall\(\)](#)

[hashtab\\_funall\(\)](#)

**38.6.3.10 atomarray\_getatoms()**

```
t_max_err atomarray_getatoms (
    t_atomarray * x,
    long * ac,
    t_atom ** av )
```

Retrieve a pointer to the first atom in the internal array of atoms.

This method does not copy the atoms, but simply provides access to them. To retrieve a copy of the atoms use [atomarray\\_copyatoms\(\)](#).

**Parameters**

x	The atomarray instance.
ac	The address of a long where the number of atoms will be set.
av	The address of a <a href="#">t_atom</a> pointer where the address of the first atom of the array will be set.

**Returns**

A Max error code.

**See also**

[atomarray\\_copyatoms\(\)](#)

**38.6.3.11 atomarray\_getflags()**

```
long atomarray_getflags (
    t_atomarray * x )
```

Get the atomarray flags.

**Parameters**

x	The atomarray instance.
---	-------------------------

**Returns**

The current value of the atomarray flags.

**38.6.3.12 atomarray\_getindex()**

```
t_max_err atomarray_getindex (
    t_atomarray * x,
    long index,
    t_atom * av )
```

Copy an a specific atom from the array.

**Parameters**

x	The atomarray instance.
index	The zero-based index into the array from which to retrieve an atom pointer.
av	The address of an atom to contain the copy.

**Returns**

A Max error code.

**Remarks****Example:**

```
{
    t_atom a;
    // fetch a copy of the second atom in a previously existing array
    atomarray_getindex(anAtomarray, 1, &a);
    // do something with the atom here...
}
```

**38.6.3.13 atomarray\_getsize()**

```
t_atom_long atomarray_getsize (
    t_atomarray * x )
```

Return the number of atoms in the array.

**Parameters**

x	The atomarray instance.
---	-------------------------

**Returns**

The number of atoms in the array.

**38.6.3.14 atomarray\_new()**

```
BEGIN_USING_C_LINKAGE t_atomarray* atomarray_new (
    long ac,
    t_atom * av )
```

Create a new atomarray object.

Note that atoms provided to this function will be *copied*. The copies stored internally to the atomarray instance. You can free the atomarray by calling [object\\_free\(\)](#).

**Parameters**

ac	The number of atoms to be initially contained in the atomarray.
av	A pointer to the first of an array of atoms to initially copy into the atomarray.

**Returns**

Pointer to the new atomarray object.

**Remarks**

Note that due to the unusual prototype of this method that you cannot instantiate this object using the [object\\_new\\_typed\(\)](#) function. If you wish to use the dynamically bound creator to instantiate the object, you should instead should use [object\\_new\(\)](#) as demonstrated below. The primary reason that you might choose to instantiate an atomarray using [object\\_new\(\)](#) instead of [atomarray\\_new\(\)](#) is for using the atomarray object in code that is also intended to run in Max 4.

```
object_new(CLASS_NOBOX, gensym("atomarray"), argc, argv);
```

**See also**

[atomarray\\_duplicate\(\)](#)

### 38.6.3.15 atomarray\_setatoms()

```
t_max_err atomarray_setatoms (
    t_atomarray * x,
    long ac,
    t_atom * av )
```

Replace the existing array contents with a new set of atoms Note that atoms provided to this function will be *copied*.

The copies stored internally to the atomarray instance.

#### Parameters

<i>x</i>	The atomarray instance.
<i>ac</i>	The number of atoms to be initially contained in the atomarray.
<i>av</i>	A pointer to the first of an array of atoms to initially copy into the atomarray.

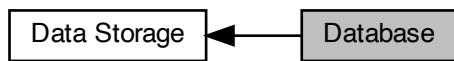
#### Returns

A Max error code.

## 38.7 Database

Max's database ( i.e.

Collaboration diagram for Database:



## Typedefs

- **typedef t\_object t\_database**  
*A database object.*
- **typedef t\_object t\_db\_result**  
*A database result object.*
- **typedef t\_object t\_db\_view**  
*A database view object.*

## Functions

- **BEGIN\_USING\_C\_LINKAGE t\_max\_err db\_open (t\_symbol \*dbname, const char \*fullpath, t\_database \*\*db)**  
*Create an instance of a database.*
- **t\_max\_err db\_open\_ext (t\_symbol \*dbname, const char \*fullpath, t\_database \*\*db, long flags)**  
*Create an instance of a database.*
- **t\_max\_err db\_close (t\_database \*\*db)**  
*Close an open database.*
- **t\_max\_err db\_query (t\_database \*db, t\_db\_result \*\*dbresult, const char \*sql,...)**  
*Execute a SQL query on the database.*
- **t\_max\_err db\_vquery (t\_database \*db, t\_db\_result \*\*dbresult, const char \*s, va\_list ap)**  
*Execute a SQL query on the database.*
- **t\_max\_err db\_query\_direct (t\_database \*db, t\_db\_result \*\*dbresult, const char \*sql)**  
*Execute a SQL query on the database.*
- **t\_max\_err db\_query\_silent (t\_database \*db, t\_db\_result \*\*dbresult, const char \*sql,...)**  
*Execute a SQL query on the database, temporarily overriding the database's error logging attribute.*
- **t\_max\_err db\_query\_getlastinsertid (t\_database \*db, long \*id)**  
*Determine the id (key) number for the most recent INSERT query executed on the database.*
- **t\_max\_err db\_query\_table\_new (t\_database \*db, const char \*tablename)**  
*Create a new table in a database.*
- **t\_max\_err db\_query\_table\_addcolumn (t\_database \*db, const char \*tablename, const char \*columnname, const char \*columntype, const char \*flags)**  
*Add a new column to an existing table in a database.*
- **t\_max\_err db\_transaction\_start (t\_database \*db)**  
*Begin a database transaction.*
- **t\_max\_err db\_transaction\_end (t\_database \*db)**  
*Finalize a database transaction.*
- **t\_max\_err db\_transaction\_flush (t\_database \*db)**  
*Force any open transactions to close.*
- **t\_max\_err db\_view\_create (t\_database \*db, const char \*sql, t\_db\_view \*\*dbview)**  
*A database view is a way of looking at a particular set of records in the database.*
- **t\_max\_err db\_view\_remove (t\_database \*db, t\_db\_view \*\*dbview)**  
*Remove a database view created using db\_view\_create().*
- **t\_max\_err db\_view\_getresult (t\_db\_view \*dbview, t\_db\_result \*\*result)**  
*Fetch the pointer for a t\_db\_view's query result.*
- **t\_max\_err db\_view\_setquery (t\_db\_view \*dbview, char \*newquery)**  
*Set the query used by the view.*
- **char \*\* db\_result\_nextrecord (t\_db\_result \*result)**  
*Return the next record from a set of results that you are walking.*
- **void db\_result\_reset (t\_db\_result \*result)**  
*Reset the interface for walking a result's record list to the first record.*
- **void db\_result\_clear (t\_db\_result \*result)**  
*Zero-out a database result.*
- **long db\_result\_numrecords (t\_db\_result \*result)**  
*Return a count of all records in the query result.*
- **long db\_result\_numfields (t\_db\_result \*result)**  
*Return a count of all fields (columns) in the query result.*

- `char * db_result_fieldname (t_db_result *result, long fieldindex)`  
*Return the name of a field specified by its index number.*
- `char * db_result_string (t_db_result *result, long recordindex, long fieldindex)`  
*Return a single value from a result according to its index and field coordinates.*
- `long db_result_long (t_db_result *result, long recordindex, long fieldindex)`  
*Return a single value from a result according to its index and field coordinates.*
- `float db_result_float (t_db_result *result, long recordindex, long fieldindex)`  
*Return a single value from a result according to its index and field coordinates.*
- `t_ptr_uint db_result_datetimeinseconds (t_db_result *result, long recordindex, long fieldindex)`  
*Return a single value from a result according to its index and field coordinates.*
- `void db_util_stringtoseconds (const char *string, t_ptr_uint *date)`  
*A utility to convert from a sql datetime string into seconds.*
- `void db_util_datetostring (const t_ptr_uint date, char *string)`  
*A utility to convert from seconds into a sql-ready datetime string.*

### 38.7.1 Detailed Description

Max's database ( i.e.

`t_database` ) support currently consists of a SQLite ( <http://sqlite.org> ) extension which is loaded dynamically by Max at launch time. Because it is loaded dynamically, all interfacing with the sqlite object relies on Max's message passing interface, using `object_method()` and related functions.

For most common database needs, a C-interface is defined in the `ext_database.h` header file and implemented in the `ext_database.c` source file. The functions defined in this interface wrap the message passing calls and provide a convenient means by which you can work with databases. `ext_database.c` is located in the 'common' folder inside of the 'max-includes' folder. If you use any of the functions defined `ext_database.h`, you will need to add `ext_database.c` to your project.

### 38.7.2 Typedef Documentation

#### 38.7.2.1 `t_database`

```
typedef t_object t_database
```

A database object.

Use `db_open()` and `db_close()` to create and free database objects.

#### 38.7.2.2 `t_db_result`

```
typedef t_object t_db_result
```

A database result object.

This is what the database object returns when a query is executed.

### 38.7.2.3 `t_db_view`

```
typedef t_object t_db_view
```

A database view object.

A database view wraps a query and a result for a given database, and is always updated and in-sync with the database.

## 38.7.3 Function Documentation

### 38.7.3.1 `db_close()`

```
t_max_err db_close (
    t_database ** db )
```

Close an open database.

#### Parameters

<code>db</code>	The address of the <code>t_database</code> pointer for your database instance. The pointer will be freed and set NULL upon return.
-----------------	--

#### Returns

An error code.

### 38.7.3.2 `db_open()`

```
BEGIN_USING_C_LINKAGE t_max_err db_open (
    t_symbol * dbname,
    const char * fullpath,
    t_database ** db )
```

Create an instance of a database.

#### Parameters

<code>dbname</code>	The name of the database.
<code>fullpath</code>	If a database with this dbname is not already open, this will specify a full path to the location where the database is stored on disk. If NULL is passed for this argument, the database will reside in memory only. The path should be formatted as a Max style path.
<code>db</code>	The address of a <code>t_database</code> pointer that will be set to point to the new database instance. If the pointer is not NULL, then it will be treated as a pre-existing database instance and thus will be freed.

**Returns**

An error code.

**38.7.3.3 db\_open\_ext()**

```
t_max_err db_open_ext (
    t_symbol * dbname,
    const char * fullpath,
    t_database ** db,
    long flags )
```

Create an instance of a database.

**Parameters**

<i>dbname</i>	The name of the database.
<i>fullpath</i>	If a database with this dbname is not already open, this will specify a full path to the location where the database is stored on disk. If NULL is passed for this argument, the database will reside in memory only. The path should be formatted as a Max style path.
<i>db</i>	The address of a <a href="#">t_database</a> pointer that will be set to point to the new database instance. If the pointer is not NULL, then it will be treated as a pre-existing database instance and thus will be freed.
<i>flags</i>	Any flags to be passed to the database backend while opening the db. At this time, DB_OPEN_FLAGS_READONLY (0x01) is the only flag available.

**Returns**

An error code.

**38.7.3.4 db\_query()**

```
t_max_err db_query (
    t_database * db,
    t_db_result ** dbresult,
    const char * sql,
    ... )
```

Execute a SQL query on the database.

**Parameters**

<i>db</i>	The <a href="#">t_database</a> pointer for your database instance.
<i>dbresult</i>	The address of a <a href="#">t_db_result</a> pointer. If the pointer is passed-in set to NULL then a new dbresult will be created. If the pointer is not NULL then it is assumed to be a valid dbresult, which will be filled in with the query results. When you are done with the dbresult you should free it with <a href="#">object_free()</a> .
<i>sql</i>	A C-string containing a valid SQL query, possibly with sprintf() formatting codes.
...	If an sprintf() formatting codes are used in the sql string, these values will be interpolated into the sql string.

Cycling '74

**Returns**

An error code.

**38.7.3.5 db\_query\_direct()**

```
t_max_err db_query_direct (
    t_database * db,
    t_db_result ** dbresult,
    const char * sql )
```

Execute a SQL query on the database.

**Parameters**

<i>db</i>	The <a href="#">t_database</a> pointer for your database instance.
<i>dbresult</i>	The address of a <a href="#">t_db_result</a> pointer. If the pointer is passed-in set to NULL then a new dbresult will be created. If the pointer is not NULL then it is assumed to be a valid dbresult, which will be filled in with the query results. When you are done with the dbresult you should free it with <a href="#">object_free()</a> .
<i>sql</i>	A C-string containing a valid SQL query.

**Returns**

An error code.

**38.7.3.6 db\_query\_getlastinsertid()**

```
t_max_err db_query_getlastinsertid (
    t_database * db,
    long * id )
```

Determine the id (key) number for the most recent INSERT query executed on the database.

**Parameters**

<i>db</i>	The <a href="#">t_database</a> pointer for your database instance.
<i>id</i>	The address of a variable to hold the result on return.

**Returns**

An error code.

### 38.7.3.7 db\_query\_silent()

```
t_max_err db_query_silent (
    t_database * db,
    t_db_result ** dbresult,
    const char * sql,
    ...
)
```

Execute a SQL query on the database, temporarily overriding the database's error logging attribute.

#### Parameters

<i>db</i>	The <a href="#">t_database</a> pointer for your database instance.
<i>dbresult</i>	The address of a <a href="#">t_db_result</a> pointer. If the pointer is passed-in set to NULL then a new dbresult will be created. If the pointer is not NULL then it is assumed to be a valid dbresult, which will be filled in with the query results. When you are done with the dbresult you should free it with <a href="#">object_free()</a> .
<i>sql</i>	A C-string containing a valid SQL query, possibly with sprintf() formatting codes.
...	If an sprintf() formatting codes are used in the sql string, these values will be interpolated into the sql string.

#### Returns

An error code.

### 38.7.3.8 db\_query\_table\_addcolumn()

```
t_max_err db_query_table_addcolumn (
    t_database * db,
    const char * tablename,
    const char * columnname,
    const char * columntype,
    const char * flags )
```

Add a new column to an existing table in a database.

#### Parameters

<i>db</i>	The <a href="#">t_database</a> pointer for your database instance.
<i>tablename</i>	The name of the table to which the column should be added.
<i>columnname</i>	The name to use for the new column.
<i>columntype</i>	The SQL type for the data that will be stored in the column. For example: "INTEGER" or "VARCHAR"
<i>flags</i>	If you wish to specify any additional information for the column, then pass that here. Otherwise pass NULL.

**Returns**

An error code.

**38.7.3.9 db\_query\_table\_new()**

```
t_max_err db_query_table_new (
    t_database * db,
    const char * tablename )
```

Create a new table in a database.

**Parameters**

<i>db</i>	The <a href="#">t_database</a> pointer for your database instance.
<i>tablename</i>	The name to use for the new table. The new table will be created with one column, which holds the primary key for the table, and is named according the form {tablename}_id.

**Returns**

An error code.

**38.7.3.10 db\_result\_clear()**

```
void db_result_clear (
    t_db_result * result )
```

Zero-out a database result.

**Parameters**

<i>result</i>	The <a href="#">t_db_result</a> pointer for your query results.
---------------	---

**38.7.3.11 db\_result\_datetimeminseconds()**

```
t_ptr_uint db_result_datetimeminseconds (
    t_db_result * result,
    long recordindex,
    long fieldindex )
```

Return a single value from a result according to its index and field coordinates.

The value will be coerced from an expected datetime field into seconds.

#### Parameters

<i>result</i>	The <a href="#">t_db_result</a> pointer for your query results.
<i>recordindex</i>	The zero-based index number of the record (row) in the result.
<i>fieldindex</i>	The zero-based index number of the field (column) in the result.

#### Returns

The datetime represented in seconds.

### 38.7.3.12 db\_result\_fieldname()

```
char* db_result_fieldname (
    t_db_result * result,
    long fieldindex )
```

Return the name of a field specified by its index number.

#### Parameters

<i>result</i>	The <a href="#">t_db_result</a> pointer for your query results.
<i>fieldindex</i>	The zero-based index number of the field (column) in the result.

#### Returns

A C-String with the name of the field.

### 38.7.3.13 db\_result\_float()

```
float db_result_float (
    t_db_result * result,
    long recordindex,
    long fieldindex )
```

Return a single value from a result according to its index and field coordinates.

**Parameters**

<i>result</i>	The <a href="#">t_db_result</a> pointer for your query results.
<i>recordindex</i>	The zero-based index number of the record (row) in the result.
<i>fieldindex</i>	The zero-based index number of the field (column) in the result.

**Returns**

The content of the specified cell from the result scanned out to a float.

**38.7.3.14 db\_result\_long()**

```
long db_result_long (
    t\_db\_result * result,
    long recordindex,
    long fieldindex )
```

Return a single value from a result according to its index and field coordinates.

**Parameters**

<i>result</i>	The <a href="#">t_db_result</a> pointer for your query results.
<i>recordindex</i>	The zero-based index number of the record (row) in the result.
<i>fieldindex</i>	The zero-based index number of the field (column) in the result.

**Returns**

The content of the specified cell from the result scanned out to a long int.

**38.7.3.15 db\_result\_nextrecord()**

```
char** db_result_nextrecord (
    t\_db\_result * result )
```

Return the next record from a set of results that you are walking.

When you are returned a result from a query of the database, the result is prepared for walking the results from the beginning. You can also reset the result manually to the beginning of the record list by calling [db\\_result\\_reset\(\)](#).

**Parameters**

<i>result</i>	The <a href="#">t_db_result</a> pointer for your query results.
---------------	---

**Returns**

An array of C-Strings with the values for every requested column (field) of a database record. To find out how many columns are represented in the array, use [db\\_result\\_numfields\(\)](#).

**38.7.3.16 db\_result\_numfields()**

```
long db_result_numfields (
    t_db_result * result )
```

Return a count of all fields (columns) in the query result.

**Parameters**

<i>result</i>	The <a href="#">t_db_result</a> pointer for your query results.
---------------	---

**Returns**

The count of fields in the query result.

**38.7.3.17 db\_result\_numrecords()**

```
long db_result_numrecords (
    t_db_result * result )
```

Return a count of all records in the query result.

**Parameters**

<i>result</i>	The <a href="#">t_db_result</a> pointer for your query results.
---------------	---

**Returns**

The count of records in the query result.

### 38.7.3.18 db\_result\_reset()

```
void db_result_reset (
    t_db_result * result )
```

Reset the interface for walking a result's record list to the first record.

#### Parameters

<i>result</i>	The <a href="#">t_db_result</a> pointer for your query results.
---------------	---

### 38.7.3.19 db\_result\_string()

```
char* db_result_string (
    t_db_result * result,
    long recordindex,
    long fieldindex )
```

Return a single value from a result according to its index and field coordinates.

#### Parameters

<i>result</i>	The <a href="#">t_db_result</a> pointer for your query results.
<i>recordindex</i>	The zero-based index number of the record (row) in the result.
<i>fieldindex</i>	The zero-based index number of the field (column) in the result.

#### Returns

A C-String with the content of the specified cell in the result.

### 38.7.3.20 db\_transaction\_end()

```
t_max_err db_transaction_end (
    t_database * db )
```

Finalize a database transaction.

#### Parameters

<i>db</i>	The <a href="#">t_database</a> pointer for your database instance.
-----------	--

**Returns**

An error code.

**38.7.3.21 db\_transaction\_flush()**

```
t_max_err db_transaction_flush (
    t_database * db )
```

Force any open transactions to close.

**Parameters**

<code>db</code>	The <a href="#">t_database</a> pointer for your database instance.
-----------------	--

**Returns**

An error code.

**38.7.3.22 db\_transaction\_start()**

```
t_max_err db_transaction_start (
    t_database * db )
```

Begin a database transaction.

When you are working with a file-based database, then the database will not be flushed to disk until `db_transacation_end()` is called. This means that you can much more efficiently execute a sequence of queries in one transaction rather than independently.

That database object reference counts transactions, so it is possible nest calls to `db_transacation_start()` and `db_transacation_end()`. It is important to balance all calls with `db_transacation_end()` or the database contents will never be flushed to disk.

**Parameters**

<code>db</code>	The <a href="#">t_database</a> pointer for your database instance.
-----------------	--

**Returns**

An error code.

### 38.7.3.23 db\_util\_datetostring()

```
void db_util_datetostring (
    const t_ptr_uint date,
    char * string )
```

A utility to convert from seconds into a sql-ready datetime string.

#### Parameters

<i>date</i>	The datetime represented in seconds.
<i>string</i>	The address of a valid C-string whose contents will be set to a SQL-ready string format upon return.

### 38.7.3.24 db\_util\_stringtodate()

```
void db_util_stringtodate (
    const char * string,
    t_ptr_uint * date )
```

A utility to convert from a sql datetime string into seconds.

#### Parameters

<i>string</i>	A C-string containing a date and time in SQL format.
<i>date</i>	The datetime represented in seconds upon return.

### 38.7.3.25 db\_view\_create()

```
t_max_err db_view_create (
    t_database * db,
    const char * sql,
    t_db_view ** dbview )
```

A database view is a way of looking at a particular set of records in the database.

This particular set of records is defined with a standard SQL query, and the view maintains a copy of the results of the query internally. Any time the database is modified the internal result set is updated, and any objects listening to the view are notified via [object\\_notify\(\)](#).

#### Parameters

<i>db</i>	The <a href="#">t_database</a> pointer for your database instance.
<i>sql</i>	A SQL query that defines the set of results provided by the view.
<i>dbview</i>	The address of a NULL <a href="#">t_db_view</a> pointer which will be set with the new view upon return.

**Returns**

An error code.

**38.7.3.26 db\_view\_getresult()**

```
t_max_err db_view_getresult (
    t_db_view * dbview,
    t_db_result ** result )
```

Fetch the pointer for a [t\\_db\\_view](#)'s query result.

**Parameters**

<i>dbview</i>	The <a href="#">t_db_view</a> pointer for your database view instance.
<i>result</i>	The address of a pointer to a <a href="#">t_db_result</a> object. This pointer will be overwritten with the view's result pointer upon return.

**Returns**

An error code.

**38.7.3.27 db\_view\_remove()**

```
t_max_err db_view_remove (
    t_database * db,
    t_db_view ** dbview )
```

Remove a database view created using [db\\_view\\_create\(\)](#).

**Parameters**

<i>db</i>	The <a href="#">t_database</a> pointer for your database instance for which this view was created.
<i>dbview</i>	The address of the <a href="#">t_db_view</a> pointer for the view. This pointer will be freed and set NULL upon return.

**Returns**

An error code.

### 38.7.3.28 db\_view\_setquery()

```
t_max_err db_view_setquery (
    t_db_view * dbview,
    char * newquery )
```

Set the query used by the view.

#### Parameters

<i>dbview</i>	The <a href="#">t_db_view</a> pointer for your database view instance.
<i>newquery</i>	The SQL string to define a new query for the view, replacing the old query.

#### Returns

An error code.

### 38.7.3.29 db\_vquery()

```
t_max_err db_vquery (
    t_database * db,
    t_db_result ** dbresult,
    const char * s,
    va_list ap )
```

Execute a SQL query on the database.

#### Parameters

<i>db</i>	The <a href="#">t_database</a> pointer for your database instance.
<i>dbresult</i>	The address of a <a href="#">t_db_result</a> pointer. If the pointer is passed-in set to NULL then a new dbresult will be created. If the pointer is not NULL then it is assumed to be a valid dbresult, which will be filled in with the query results. When you are done with the dbresult you should free it with <a href="#">object_free()</a> .
<i>sql</i>	A C-string containing a valid SQL query, possibly with sprintf() formatting codes.
<i>ap</i>	va_list containing any additional arguments necessary for sprintf()-format processing of the sql string.

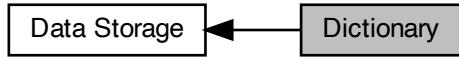
#### Returns

An error code.

## 38.8 Dictionary

Max 5, introduced the [t\\_dictionary](#) structure/object.

Collaboration diagram for Dictionary:



## Data Structures

- struct `t_dictionary_entry`  
*A dictionary entry.*
- struct `t_dictionary`  
*The dictionary object.*

## Functions

- `t_dictionary * dictionary_new (void)`  
*Create a new dictionary object.*
- `t_max_err dictionary_appendlong (t_dictionary *d, t_symbol *key, t_atom_long value)`  
*Add a long integer value to the dictionary.*
- `t_max_err dictionary_appendfloat (t_dictionary *d, t_symbol *key, double value)`  
*Add a double-precision float value to the dictionary.*
- `t_max_err dictionary_appendsym (t_dictionary *d, t_symbol *key, t_symbol *value)`  
*Add a `t_symbol*` value to the dictionary.*
- `t_max_err dictionary_appendatom (t_dictionary *d, t_symbol *key, t_atom *value)`  
*Add a `t_atom*` value to the dictionary.*
- `t_max_err dictionary_appendstring (t_dictionary *d, t_symbol *key, const char *value)`  
*Add a C-string to the dictionary.*
- `t_max_err dictionary_appendatoms (t_dictionary *d, t_symbol *key, long argc, t_atom *argv)`  
*Add an array of atoms to the dictionary.*
- `t_max_err dictionary_appendatomarray (t_dictionary *d, t_symbol *key, t_object *value)`  
*Add an Atom Array object to the dictionary.*
- `t_max_err dictionary_appenddictionary (t_dictionary *d, t_symbol *key, t_object *value)`  
*Add a dictionary object to the dictionary.*
- `t_max_err dictionary_appendobject (t_dictionary *d, t_symbol *key, t_object *value)`  
*Add an object to the dictionary.*
- `t_max_err dictionary_getlong (C74_CONST t_dictionary *d, t_symbol *key, t_atom_long *value)`  
*Retrieve a long integer from the dictionary.*
- `t_max_err dictionary_getfloat (C74_CONST t_dictionary *d, t_symbol *key, double *value)`  
*Retrieve a double-precision float from the dictionary.*
- `t_max_err dictionary_getsym (C74_CONST t_dictionary *d, t_symbol *key, t_symbol **value)`  
*Retrieve a `t_symbol*` from the dictionary.*

- `t_max_err dictionary_getatom (C74_CONST t_dictionary *d, t_symbol *key, t_atom *value)`  
*Copy a `t_atom` from the dictionary.*
- `t_max_err dictionary_getstring (C74_CONST t_dictionary *d, t_symbol *key, const char **value)`  
*Retrieve a C-string pointer from the dictionary.*
- `t_max_err dictionary_getatoms (C74_CONST t_dictionary *d, t_symbol *key, long *argc, t_atom **argv)`  
*Retrieve the address of a `t_atom` array of in the dictionary.*
- `t_max_err dictionary_getatoms_ext (const t_dictionary *d, t_symbol *key, long stringstosymbols, long *argc, t_atom **argv)`  
*Retrieve the address of a `t_atom` array in the dictionary.*
- `t_max_err dictionary_copyatoms (C74_CONST t_dictionary *d, t_symbol *key, long *argc, t_atom **argv)`  
*Retrieve copies of a `t_atom` array in the dictionary.*
- `t_max_err dictionary_getatomarray (C74_CONST t_dictionary *d, t_symbol *key, t_object **value)`  
*Retrieve a `t_atomarray` pointer from the dictionary.*
- `t_max_err dictionary_getdictionary (C74_CONST t_dictionary *d, t_symbol *key, t_object **value)`  
*Retrieve a `t_dictionary` pointer from the dictionary.*
- `t_max_err dictionary_get_ex (t_dictionary *d, t_symbol *key, long *ac, t_atom **av, char *errstr)`  
*Retrieve the address of a `t_atom` array of in the dictionary within nested dictionaries.*
- `t_max_err dictionary_getobject (C74_CONST t_dictionary *d, t_symbol *key, t_object **value)`  
*Retrieve a `t_object` pointer from the dictionary.*
- `long dictionary_entryisstring (C74_CONST t_dictionary *d, t_symbol *key)`  
*Test a key to set if the data stored with that key contains a `t_string` object.*
- `long dictionary_entryisatomarray (C74_CONST t_dictionary *d, t_symbol *key)`  
*Test a key to set if the data stored with that key contains a `t_atomarray` object.*
- `long dictionary_entryisdictionary (C74_CONST t_dictionary *d, t_symbol *key)`  
*Test a key to set if the data stored with that key contains a `t_dictionary` object.*
- `long dictionary_hasentry (C74_CONST t_dictionary *d, t_symbol *key)`  
*Test a key to set if it exists in the dictionary.*
- `t_atom_long dictionary_getentrycount (C74_CONST t_dictionary *d)`  
*Return the number of keys in a dictionary.*
- `t_max_err dictionary_getkeys (C74_CONST t_dictionary *d, long *numkeys, t_symbol ***keys)`  
*Retrieve all of the key names stored in a dictionary.*
- `void dictionary_freekeys (t_dictionary *d, long numkeys, t_symbol **keys)`  
*Free memory allocated by the `dictionary_getkeys()` method.*
- `t_max_err dictionary_deleteentry (t_dictionary *d, t_symbol *key)`  
*Remove a value from the dictionary.*
- `t_max_err dictionary_chuckentry (t_dictionary *d, t_symbol *key)`  
*Remove a value from the dictionary without freeing it.*
- `t_max_err dictionary_clear (t_dictionary *d)`  
*Delete all values from a dictionary.*
- `void dictionary_funall (t_dictionary *d, method fun, void *arg)`  
*Call the specified function for every entry in the dictionary.*
- `t_symbol * dictionary_entry_getkey (t_dictionary_entry *x)`  
*Given a `t_dictionary_entry*`, return the key associated with that entry.*
- `void dictionary_entry_getvalue (t_dictionary_entry *x, t_atom *value)`  
*Given a `t_dictionary_entry*`, return the value associated with that entry.*
- `void dictionary_entry_getvalues (t_dictionary_entry *x, long *argc, t_atom **argv)`  
*Given a `t_dictionary_entry*`, return the values associated with that entry.*

- `t_max_err dictionary_copyunique (t_dictionary *d, t_dictionary *copyfrom)`  
*Given 2 dictionaries, copy the keys unique to one of the dictionaries to the other dictionary.*
- `t_max_err dictionary_getdeflong (const t_dictionary *d, t_symbol *key, t_atom_long *value, t_atom_long def)`  
*Retrieve a long integer from the dictionary.*
- `t_max_err dictionary_getdeffloat (const t_dictionary *d, t_symbol *key, double *value, double def)`  
*Retrieve a double-precision float from the dictionary.*
- `t_max_err dictionary_getdefsym (const t_dictionary *d, t_symbol *key, t_symbol **value, t_symbol *def)`  
*Retrieve a `t_symbol`\* from the dictionary.*
- `t_max_err dictionary_getdefatom (const t_dictionary *d, t_symbol *key, t_atom *value, t_atom *def)`  
*Retrieve a `t_atom`\* from the dictionary.*
- `t_max_err dictionary_getdefstring (const t_dictionary *d, t_symbol *key, const char **value, char *def)`  
*Retrieve a C-string from the dictionary.*
- `t_max_err dictionary_getdefatoms (t_dictionary *d, t_symbol *key, long *argc, t_atom **argv, t_atom *def)`  
*Retrieve the address of a `t_atom` array of in the dictionary.*
- `t_max_err dictionary_copydefatoms (t_dictionary *d, t_symbol *key, long *argc, t_atom **argv, t_atom *def)`  
*Retrieve copies of a `t_atom` array in the dictionary.*
- `t_max_err dictionary_dump (t_dictionary *d, long recurse, long console)`  
*Print the contents of a dictionary to the Max window.*
- `t_max_err dictionary_copyentries (t_dictionary *src, t_dictionary *dst, t_symbol **keys)`  
*Copy specified entries from one dictionary to another.*
- `t_dictionary * dictionary_sprintf (C74_CONST char *fmt,...)`  
*Create a new dictionary populated with values using a combination of attribute and sprintf syntax.*
- `long dictionary_transaction_lock (t_dictionary *d)`  
*Take a lock on a dictionary for preventing dictionary lock for transactions across multiple calls, or holding on to internal dictionary element pointers for complex operations.*
- `long dictionary_transaction_unlock (t_dictionary *d)`  
*Release a lock on a dictionary for preventing dictionary lock for transactions across multiple calls, or holding on to internal dictionary element pointers for complex operations.*
- `t_max_err dictionary_read (const char *filename, short path, t_dictionary **d)`  
*Read the specified JSON file and return a `t_dictionary` object.*
- `t_max_err dictionary_write (t_dictionary *d, const char *filename, short path)`  
*Serialize the specified `t_dictionary` object to a JSON file.*
- `t_max_err dictionary_read_yaml (const char *filename, const short path, t_dictionary **d)`  
*Read the specified YAML file and return a `t_dictionary` object.*
- `t_max_err dictionary_write_yaml (const t_dictionary *d, const char *filename, const short path)`  
*Serialize the specified `t_dictionary` object to a YAML file.*
- `void postdictionary (t_object *d)`  
*Print the contents of a dictionary to the Max window.*

### 38.8.1 Detailed Description

Max 5, introduced the `t_dictionary` structure/object.

This is used for object prototypes, object serialization, object constructors, and many other tasks. A dictionary is ultimately a collection of atom values assigned to symbolic keys. In addition to primitive `A_LONG`, `A_FLOAT`, and `A_SYM` atom types, the `A_OBJ` atom type is used for `t_atomarray` (for a set of atoms assigned to a key), `t_dictionary` (for hierarchical use), `t_string` (for large blocks of text which we don't wish to bloat the symbol table), and potentially other

object data types. Internally, the dictionary object uses a combination data structure of a hash table (for fast key lookup) and a linked-list (to maintain ordering of information within the dictionary).

Dictionaries are clonable entities, but note that all the member objects of a given dictionary may not be clonable. At the time of this writing, for example, the `t_string` object is not clonable, though it will be made clonable in the near future. In order for prototype entities to be guaranteed their passage into the constructor, they must be clonable (currently a symbol conversion is in place for the `t_string` class).

## 38.8.2 Using Dictionaries

Dictionaries are used in many places in Max 5. They can be confusing in many respects. It is easy to produce memory leaks or bugs where objects are freed twice. It is easy to confuse what type of dictionary is used for what. This page will begin with some high level information to help understand when to free and when not to free. Then, we will offer recipes for using dictionaries to accomplish common tasks.

### 38.8.2.1 Understanding Dictionaries

A dictionary stores atom values under named key entries. These atoms can contain `A_OBJ` values. When the dictionary is freed, any `A_OBJ` values that are in the dictionary will also be freed. Thus, it is easy to mistakenly free objects twice, thus this is something to be careful about. For example, look at this code:

```
t_dictionary *d = dictionary_new();
t_dictionary *sd = dictionary_new();
dictionary_appenddictionary(d, gensym("subdictionary"), sd);
do_something(d);
object_free(d); // this will free *both* d and sd since sd is contained by d
// freeing "sd" here would be bad
```

You primarily need to keep this in mind when calling `dictionary_appendobject()`, `dictionary_appenddictionary()`, or `dictionary_appendatomarray()`. So, what do you do if you need to free a dictionary but you also want to hang on to an object that is inside of the dictionary? In this case, chuck the entry in question first. For example, let's assume that for some reason you cannot free the "sd" dictionary in the code above. Perhaps it doesn't belong to you. But, to do some operation you need to append it to a new dictionary. Then, do this:

```
void function_foo(t_dictionary *sd) {
    t_dictionary *d = dictionary_new();
    dictionary_appenddictionary(d, gensym("subdictionary"), sd);
    do_something(d);
    dictionary_chuckentry(d, gensym("subdictionary"));
    object_free(d);
}
```

### 38.8.2.2 When to Free a Dictionary

So, how do you know when you need to free a dictionary? Well, generally if you make a dictionary, you need to free it when you are done (unless you transfer ownership of the dictionary to someone else). On the other hand, if you are passed a dictionary (i.e. as a parameter of your function or method) then it is not yours to free and you should just use it. However, it is not always obvious that you made a dictionary vs just borrowed it.

Here are some common (and not so common) ways to make a dictionary. These functions return a new dictionary and thus the dictionary you get should be freed when you are done, unless you pass the dictionary on to someone else who will free it at an appropriate time. Here they are:

- `dictionary_new()`

- `dictionary_clone()`
- [dictionary\\_read\(\)](#)
- [dictionary\\_sprintf\(\)](#)
- `dictionary_vsprintf()`
- `jsonreader_parse()`
- `jpatcher_monikerforobject()`
- `class_cloneprototype()`
- `prototype_getdictionary()`
- `clipboard_todictionary()`
- `jpatchercontroller_copytodictionary()`
- probably others of course

Here are some functions that return borrowed dictionaries. These are dictionaries that you can use but you cannot free since you do not own them. Here they are:

- `dictionary_prototypefromclass()`
- `object_refpage_get_class_info_fromclassname()`
- `object_refpage_get_class_info()`
- [object\\_dictionaryarg\(\)](#)

Finally, most functions that accept dictionaries as parameters will not assume ownership of the dictionary. Usually the way ownership is assumed is if you add a dictionary as a subdictionary to a dictionary that you do not own. One exception is the utility `newobject_fromdictionary_delete()` who's name makes it clear that the dictionary will be deleted after calling the function.

### 38.8.2.3 Some Common Uses of Dictionaries

You can make a patcher by passing a dictionary to [object\\_new\\_typed\(\)](#) when making a "jpatcher". Using `atom_setparse()` and [attr\\_args\\_dictionary\(\)](#) makes this relatively easy.

Use [newobject\\_sprintf\(\)](#) to programmatically make an object in a patch. Actually, you don't explicitly use a dictionary here! If you do want more control, so you can touch the dictionary to customize it, then see the next bullet.

Use [dictionary\\_sprintf\(\)](#) to make a dictionary to specify a box (i.e. specify class with @maxclass attr). Then, make another dictionary and append your box dictionary to it under the key "box" via [dictionary\\_appenddictionary\(\)](#). Finally, make your object with [newobject\\_fromdictionary\(\)](#).

See also

- [Linked List](#)
- [Hash Table](#)

Version

5.0

### 38.8.3 Reading and Writing Dictionaries as JSON

#### 38.8.3.1 Creating a Dictionary from JSON

The easiest way to get read a `t_dictionary` from a JSON file on disk is to use the `dictionary_read()` function. In some cases you may wish have more control, such as to generate JSON but not write it to disk. For that purpose you can create a `jsonreader` object as demonstrated below.

```
t_dictionary      *d = NULL;
t_max_err        err;
t_atom           result[1];
t_object         *jsonreader = (t_object*)object_new(_sym_nobox, _sym_jsonreader);
// assume we have an argument called 'jsontext' which is a const char* with the JSON
// from which we wish to create a t_dictionary instance
err = (t_max_err)object_method(jsonreader, _sym_parse, jsontext, result);
if (!err) {
    t_object *ro = (t_object*)atom_getobj(result);
    if (ro) {
        if (object_classname_compare(ro, _sym_dictionary))
            d = (t_dictionary*)ro;
        else
            object_free(ro);
    }
}
object_free(jsonreader);
// we now have a t_dictionary in d that can be used as we see fit
```

#### 38.8.3.2 Creating JSON from a Dictionary

The easiest way to get write a `t_dictionary` to disk as JSON is to use the `dictionary_write()` function. In some cases you may wish have more control, such as to generate JSON but not write it to disk. For that purpose you can create a `jsonwriter` object as demonstrated below.

```
t_object      *jsonwriter = (t_object*)object_new(_sym_nobox, _sym_jsonwriter);
t_handle     json;
const char   *str;
object_method(jsonwriter, _sym_writedictionary, d);
object_method(jsonwriter, _sym_getoutput, &json);
str = *json;
// now str contains our JSON serialization of the t_dictionary d
object_free(jsonwriter);
```

### 38.8.4 Function Documentation

#### 38.8.4.1 dictionary\_appendatom()

```
t_max_err dictionary_appendatom (
    t_dictionary * d,
    t_symbol * key,
    t_atom * value )
```

Add a `t_atom*` value to the dictionary.

##### Parameters

<code>d</code>	The dictionary instance.
<code>key</code>	The name of the key used to index the new value. All keys must be unique. If the key name already exists,
<code>Cycling '74</code>	then the existing value associated with the key will be freed prior to the new value's assignment.
<code>value</code>	The new value to append to the dictionary.

**Returns**

A Max error code.

**38.8.4.2 dictionary\_appendatomarray()**

```
t_max_err dictionary_appendatomarray (
    t_dictionary * d,
    t_symbol * key,
    t_object * value )
```

Add an [Atom Array](#) object to the dictionary.

Note that from this point on that you should not free the [t\\_atomarray\\*](#), because the atomarray is now owned by the dictionary, and freeing the dictionary will free the atomarray as discussed in [When to Free a Dictionary](#).

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The name of the key used to index the new value. All keys must be unique. If the key name already exists, then the existing value associated with the key will be freed prior to the new value's assignment.
<i>value</i>	The new value to append to the dictionary.

**Returns**

A Max error code.

**38.8.4.3 dictionary\_appendatoms()**

```
t_max_err dictionary_appendatoms (
    t_dictionary * d,
    t_symbol * key,
    long argc,
    t_atom * argv )
```

Add an array of atoms to the dictionary.

Internally these atoms will be copied into a [t\\_atomarray](#) object, which will be appended to the dictionary with the given key.

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The name of the key used to index the new value. All keys must be unique. If the key name already exists, then the existing value associated with the key will be freed prior to the new value's assignment.
<i>argc</i>	The number of atoms to append to the dictionary.
<i>argv</i>	The address of the first atom in the array to append to the dictionary.

**Returns**

A Max error code.

**38.8.4.4 dictionary\_appenddictionary()**

```
t_max_err dictionary_appenddictionary (
    t_dictionary * d,
    t_symbol * key,
    t_object * value )
```

Add a dictionary object to the dictionary.

Note that from this point on that you should not free the `t_dictionary`\* that is being added, because the newly-added dictionary is now owned by the dictionary to which it has been added, as discussed in [When to Free a Dictionary](#).

**Parameters**

<code>d</code>	The dictionary instance.
<code>key</code>	The name of the key used to index the new value. All keys must be unique. If the key name already exists, then the existing value associated with the key will be freed prior to the new value's assignment.
<code>value</code>	The new value to append to the dictionary.

**Returns**

A Max error code.

**38.8.4.5 dictionary\_appendfloat()**

```
t_max_err dictionary_appendfloat (
    t_dictionary * d,
    t_symbol * key,
    double value )
```

Add a double-precision float value to the dictionary.

**Parameters**

<code>d</code>	The dictionary instance.
<code>key</code>	The name of the key used to index the new value. All keys must be unique. If the key name already exists, then the existing value associated with the key will be freed prior to the new value's assignment.
<code>value</code>	The new value to append to the dictionary.

**Returns**

A Max error code.

**38.8.4.6 dictionary\_appendlong()**

```
t_max_err dictionary_appendlong (
    t_dictionary * d,
    t_symbol * key,
    t_atom_long value )
```

Add a long integer value to the dictionary.

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The name of the key used to index the new value. All keys must be unique. If the key name already exists, then the existing value associated with the key will be freed prior to the new value's assignment.
<i>value</i>	The new value to append to the dictionary.

**Returns**

A Max error code.

**38.8.4.7 dictionary\_appendobject()**

```
t_max_err dictionary_appendobject (
    t_dictionary * d,
    t_symbol * key,
    t_object * value )
```

Add an object to the dictionary.

Note that from this point on that you should not free the `t_object*` that is being added, because the newly-added object is now owned by the dictionary to which it has been added, as discussed in [When to Free a Dictionary](#).

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The name of the key used to index the new value. All keys must be unique. If the key name already exists, then the existing value associated with the key will be freed prior to the new value's assignment.
<i>value</i>	The new value to append to the dictionary.

**Returns**

A Max error code.

**38.8.4.8 dictionary\_appendstring()**

```
t_max_err dictionary_appendstring (
    t_dictionary * d,
    t_symbol * key,
    const char * value )
```

Add a C-string to the dictionary.

Internally this uses the `t_string` object. It is useful to use the `t_string` in dictionaries rather than the `t_symbol` to avoid bloating Max's symbol table unnecessarily.

**Parameters**

<code>d</code>	The dictionary instance.
<code>key</code>	The name of the key used to index the new value. All keys must be unique. If the key name already exists, then the existing value associated with the key will be freed prior to the new value's assignment.
<code>value</code>	The new value to append to the dictionary.

**Returns**

A Max error code.

**38.8.4.9 dictionary\_appendsym()**

```
t_max_err dictionary_appendsym (
    t_dictionary * d,
    t_symbol * key,
    t_symbol * value )
```

Add a `t_symbol`\* value to the dictionary.

**Parameters**

<code>d</code>	The dictionary instance.
<code>key</code>	The name of the key used to index the new value. All keys must be unique. If the key name already exists, then the existing value associated with the key will be freed prior to the new value's assignment.
<code>value</code>	The new value to append to the dictionary.

**Returns**

A Max error code.

### 38.8.4.10 dictionary\_chuckentry()

```
t_max_err dictionary_chuckentry (
    t_dictionary * d,
    t_symbol * key )
```

Remove a value from the dictionary without freeing it.

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to delete.

**Returns**

A max error code.

**See also**

[dictionary\\_deleteentry\(\)](#)

### 38.8.4.11 dictionary\_clear()

```
t_max_err dictionary_clear (
    t_dictionary * d )
```

Delete all values from a dictionary.

This method will free the objects in the dictionary. If freeing the objects is inappropriate or undesirable then you should iterate through the dictionary and use [dictionary\\_chuckentry\(\)](#) instead.

**Parameters**

<i>d</i>	The dictionary instance.
----------	--------------------------

**Returns**

A max error code.

**See also**

[dictionary\\_getkeys\(\)](#)  
[dictionary\\_chuckentry\(\)](#)  
[dictionary\\_deleteentry\(\)](#)

**38.8.4.12 dictionary\_copyatoms()**

```
t_max_err dictionary_copyatoms (
    C74_CONST t_dictionary * d,
    t_symbol * key,
    long * argc,
    t_atom ** argv )
```

Retrieve copies of a [t\\_atom](#) array in the dictionary.

The retrieved pointer of [t\\_atoms](#) in the dictionary has memory allocated and copied to it from within the function. You are responsible for freeing it with [sysmem\\_freeptr\(\)](#).

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>argc</i>	The address of a variable to hold the number of atoms in the array.
<i>argv</i>	The address of a variable to hold a pointer to the first atom in the array. You should initialize this pointer to NULL prior to passing it to <a href="#">dictionary_copyatoms()</a> .

**Returns**

A Max error code.

**See also**

[dictionary\\_getatoms\(\)](#)

### 38.8.4.13 dictionary\_copydefatoms()

```
t_max_err dictionary_copydefatoms (
    t_dictionary * d,
    t_symbol * key,
    long * argc,
    t_atom ** argv,
    t_atom * def )
```

Retrieve copies of a [t\\_atom](#) array in the dictionary.

The retrieved pointer of [t\\_atoms](#) in the dictionary has memory allocated and copied to it from within the function. You are responsible for freeing it with [sysmem\\_freeptr\(\)](#). If the named key doesn't exist, then copy a default array of atoms, specified as a [t\\_atomarray](#)\*.

#### Parameters

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>argc</i>	The address of a variable to hold the number of atoms in the array.
<i>argv</i>	The address of a variable to hold a pointer to the first atom in the array. You should initialize this pointer to NULL prior to passing it to <a href="#">dictionary_copyatoms()</a> .
<i>def</i>	The default values specified as an instance of the <a href="#">t_atomarray</a> object.

#### Returns

A Max error code.

#### See also

[dictionary\\_getdefatoms\(\)](#)  
[dictionary\\_copyatoms\(\)](#)

### 38.8.4.14 dictionary\_copyentries()

```
t_max_err dictionary_copyentries (
    t_dictionary * src,
    t_dictionary * dst,
    t_symbol ** keys )
```

Copy specified entries from one dictionary to another.

#### Parameters

<i>src</i>	The source dictionary from which to copy entries.
<i>dst</i>	The destination dictionary to which the entries will be copied.
<i>keys</i>	The address of the first of an array of <a href="#">t_symbol</a> * that specifies which keys to copy.

**Returns**

A Max error code.

**See also**

[dictionary\\_copyunique\(\)](#)

**38.8.4.15 dictionary\_copyunique()**

```
t_max_err dictionary_copyunique (
    t_dictionary * d,
    t_dictionary * copyfrom )
```

Given 2 dictionaries, copy the keys unique to one of the dictionaries to the other dictionary.

**Parameters**

<i>d</i>	A dictionary instance. This will be the destination for any values that are copied.
<i>copyfrom</i>	A dictionary instance from which we will copy any values with unique keys.

**Returns**

A Max error code.

**See also**

[dictionary\\_copyentries\(\)](#)

**38.8.4.16 dictionary\_deleteentry()**

```
t_max_err dictionary_deleteentry (
    t_dictionary * d,
    t_symbol * key )
```

Remove a value from the dictionary.

This method will free the object in the dictionary. If freeing the object is inappropriate or undesirable, use [dictionary\\_chuckentry\(\)](#) instead.

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to delete.

**Returns**

A max error code.

**See also**

[dictionary\\_chuckentry\(\)](#)  
[dictionary\\_clear\(\)](#)

**38.8.4.17 dictionary\_dump()**

```
t_max_err dictionary_dump (
    t_dictionary * d,
    long recurse,
    long console )
```

Print the contents of a dictionary to the Max window.

**Parameters**

<i>d</i>	The dictionary instance.
<i>recurse</i>	If non-zero, the dictionary will be recursively unravelled to the Max window. Otherwise it will only print the top level.
<i>console</i>	If non-zero, the dictionary will be posted to the console rather than the Max window. On the Mac you can view this using Console.app. On Windows you can use the free DbgView program which can be downloaded from Microsoft.

**Returns**

A Max error code.

**38.8.4.18 dictionary\_entry\_getkey()**

```
t_symbol* dictionary_entry_getkey (
    t_dictionary_entry * x )
```

Given a `t_dictionary_entry*`, return the key associated with that entry.

**Parameters**

x	The dictionary entry.
---	-----------------------

**Returns**

The key associated with the entry.

**See also**

[dictionary\\_entry\\_getvalue\(\)](#)  
[dictionary\\_funall\(\)](#)

**38.8.4.19 dictionary\_entry\_getvalue()**

```
void dictionary_entry_getvalue (
    t_dictionary_entry * x,
    t_atom * value )
```

Given a `t_dictionary_entry*`, return the value associated with that entry.

**Parameters**

x	The dictionary entry.
value	The address of a <code>t_atom</code> to which the value will be copied.

**See also**

[dictionary\\_entry\\_getkey\(\)](#)  
[dictionary\\_funall\(\)](#)

**38.8.4.20 dictionary\_entry\_getvalues()**

```
void dictionary_entry_getvalues (
    t_dictionary_entry * x,
    long * argc,
    t_atom ** argv )
```

Given a `t_dictionary_entry*`, return the values associated with that entry.

**Parameters**

<i>x</i>	The dictionary entry.
<i>argc</i>	The length of the returned <a href="#">t_atom</a> vector.
<i>argv</i>	The address of a <a href="#">t_atom</a> vector to which the values will be copied.

**See also**

[dictionary\\_entry\\_getkey\(\)](#)  
[dictionary\\_funall\(\)](#)

**38.8.4.21 dictionary\_entryisatomarray()**

```
long dictionary_entryisatomarray (
    C74_CONST t_dictionary * d,
    t_symbol * key )
```

Test a key to set if the data stored with that key contains a [t\\_atomarray](#) object.

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to test.

**Returns**

Returns true if the key contains a [t\\_atomarray](#), otherwise returns false.

**38.8.4.22 dictionary\_entryisdictionary()**

```
long dictionary_entryisdictionary (
    C74_CONST t_dictionary * d,
    t_symbol * key )
```

Test a key to set if the data stored with that key contains a [t\\_dictionary](#) object.

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to test.

**Returns**

Returns true if the key contains a [t\\_dictionary](#), otherwise returns false.

**38.8.4.23 dictionary\_entryisstring()**

```
long dictionary_entryisstring (
    C74_CONST t_dictionary * d,
    t_symbol * key )
```

Test a key to set if the data stored with that key contains a [t\\_string](#) object.

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to test.

**Returns**

Returns true if the key contains a [t\\_string](#), otherwise returns false.

**38.8.4.24 dictionary\_freekeys()**

```
void dictionary_freekeys (
    t_dictionary * d,
    long numkeys,
    t_symbol ** keys )
```

Free memory allocated by the [dictionary\\_getkeys\(\)](#) method.

**Parameters**

<i>d</i>	The dictionary instance.
<i>numkeys</i>	The address of a long where the number of keys retrieved will be set.
<i>keys</i>	The address of the first of an array <a href="#">t_symbol</a> pointers where the retrieved keys will be set.

**Returns**

A max error code.

## See also

[dictionary\\_getkeys\(\)](#)

**38.8.4.25 dictionary\_funall()**

```
void dictionary_funall (
    t_dictionary * d,
    method fun,
    void * arg )
```

Call the specified function for every entry in the dictionary.

## Parameters

<i>d</i>	The dictionary instance.
<i>fun</i>	The function to call, specified as function pointer cast to a Max #method.
<i>arg</i>	An argument that you would like to pass to the function being called.

## Remarks

The [dictionary\\_funall\(\)](#) method will call your function for every entry in the dictionary. It will pass both a pointer to the [t\\_dictionary\\_entry](#), and any argument that you provide. The following example shows a function that could be called by [dictionary\\_funall\(\)](#).

```
void my_function(t_dictionary_entry *entry, void* my_arg)
{
    t_symbol     *key;
    t_atom       value;

    key = dictionary_entry_getkey(entry);
    dictionary_entry_getvalue(entry, &value);

    // do something with key, value, and my_arg...
}
```

## See also

[dictionary\\_entry\\_getkey\(\)](#)  
[dictionary\\_entry\\_getvalue\(\)](#)

**38.8.4.26 dictionary\_get\_ex()**

```
t_max_err dictionary_get_ex (
    t_dictionary * d,
    t_symbol * key,
    long * ac,
```

```
t_atom ** av,
char * errstr )
```

Retrieve the address of a `t_atom` array of in the dictionary within nested dictionaries.

The address can index into nested dictionaries using the '::' operator. For example, the key "field::subfield" will look for the value at key "field" and then look for the value "subfield" in the value found at "field".

#### Parameters

<code>d</code>	The dictionary instance.
<code>key</code>	The key associated with the value to lookup.
<code>ac</code>	The number of return values
<code>av</code>	The return values
<code>errstr</code>	An error message if an error code was returned. Optional, pass NULL if you don't want it.

#### Returns

A Max error code.

### 38.8.4.27 dictionary\_getatom()

```
t_max_err dictionary_getatom (
    C74_CONST t_dictionary * d,
    t_symbol * key,
    t_atom * value )
```

Copy a `t_atom` from the dictionary.

#### Parameters

<code>d</code>	The dictionary instance.
<code>key</code>	The key associated with the value to lookup.
<code>value</code>	The address of variable to hold the value associated with the key.

#### Returns

A Max error code.

### 38.8.4.28 dictionary\_getatomarray()

```
t_max_err dictionary_getatomarray (
    C74_CONST t_dictionary * d,
```

```
t_symbol * key,
t_object ** value )
```

Retrieve a [t\\_atomarray](#) pointer from the dictionary.

#### Parameters

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>value</i>	The address of variable to hold the value associated with the key.

#### Returns

A Max error code.

### 38.8.4.29 dictionary\_getatoms()

```
t_max_err dictionary_getatoms (
    C74_CONST t_dictionary * d,
    t_symbol * key,
    long * argc,
    t_atom ** argv )
```

Retrieve the address of a [t\\_atom](#) array of in the dictionary.

The retrieved pointer references the [t\\_atoms](#) in the dictionary. To fetch a copy of the [t\\_atoms](#) from the dictionary, use [dictionary\\_copyatoms\(\)](#).

#### Parameters

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>argc</i>	The address of a variable to hold the number of atoms in the array.
<i>argv</i>	The address of a variable to hold a pointer to the first atom in the array.

#### Returns

A Max error code.

#### See also

[dictionary\\_copyatoms\(\)](#)  
[dictionary\\_getatoms\\_ext\(\)](#)

### 38.8.4.30 dictionary\_getatoms\_ext()

```
t_max_err dictionary_getatoms_ext (
    const t_dictionary * d,
    t_symbol * key,
    long stringstosymbols,
    long * argc,
    t_atom ** argv )
```

Retrieve the address of a [t\\_atom](#) array in the dictionary.

The retrieved pointer references the [t\\_atoms](#) in the dictionary. Optionally convert strings to symbols. To fetch a copy of the [t\\_atoms](#) from the dictionary, use [dictionary\\_copyatoms\(\)](#).

#### Parameters

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>stringstosymbols</i>	The flag to convert strings to symbols (true,false).
<i>argc</i>	The address of a variable to hold the number of atoms in the array.
<i>argv</i>	The address of a variable to hold a pointer to the first atom in the array.

#### Returns

A Max error code.

#### See also

[dictionary\\_copyatoms\(\)](#)  
[dictionary\\_getatoms\(\)](#)

### 38.8.4.31 dictionary\_getdefatom()

```
t_max_err dictionary_getdefatom (
    const t_dictionary * d,
    t_symbol * key,
    t_atom * value,
    t_atom * def )
```

Retrieve a [t\\_atom\\*](#) from the dictionary.

If the named key doesn't exist, then return a specified default value.

#### Parameters

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<small>Cycling<sup>74</sup></small> <i>value</i>	The address of variable to hold the value associated with the key.
<i>def</i>	The default value to return in the absence of the key existing in the dictionary.

**Returns**

A Max error code.

**See also**

[dictionary\\_getatom\(\)](#)

### 38.8.4.32 dictionary\_getdefatoms()

```
t_max_err dictionary_getdefatoms (
    t_dictionary * d,
    t_symbol * key,
    long * argc,
    t_atom ** argv,
    t_atom * def )
```

Retrieve the address of a [t\\_atom](#) array of in the dictionary.

The retrieved pointer references the [t\\_atoms](#) in the dictionary. To fetch a copy of the [t\\_atoms](#) from the dictionary, use [dictionary\\_copyatoms\(\)](#). If the named key doesn't exist, then return a default array of atoms, specified as a [t\\_atomarray\\*](#).

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>argc</i>	The address of a variable to hold the number of atoms in the array.
<i>argv</i>	The address of a variable to hold a pointer to the first atom in the array.
<i>def</i>	The default values specified as an instance of the <a href="#">t_atomarray</a> object.

**Returns**

A Max error code.

**See also**

[dictionary\\_getatoms\(\)](#)  
[dictionary\\_copydefatoms\(\)](#)

### 38.8.4.33 dictionary\_getdeffloat()

```
t_max_err dictionary_getdeffloat (
    const t_dictionary * d,
    t_symbol * key,
    double * value,
    double def )
```

Retrieve a double-precision float from the dictionary.

If the named key doesn't exist, then return a specified default value.

#### Parameters

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>value</i>	The address of variable to hold the value associated with the key.
<i>def</i>	The default value to return in the absence of the key existing in the dictionary.

#### Returns

A Max error code.

#### See also

[dictionary\\_getfloat\(\)](#)

### 38.8.4.34 dictionary\_getdeflong()

```
t_max_err dictionary_getdeflong (
    const t_dictionary * d,
    t_symbol * key,
    t_atom_long * value,
    t_atom_long def )
```

Retrieve a long integer from the dictionary.

If the named key doesn't exist, then return a specified default value.

#### Parameters

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>value</i>	The address of variable to hold the value associated with the key.
<i>def</i>	The default value to return in the absence of the key existing in the dictionary.

**Returns**

A Max error code.

**See also**

[dictionary\\_getlong\(\)](#)

**38.8.4.35 dictionary\_getdefstring()**

```
t_max_err dictionary_getdefstring (
    const t_dictionary * d,
    t_symbol * key,
    const char ** value,
    char * def )
```

Retrieve a C-string from the dictionary.

If the named key doesn't exist, then return a specified default value.

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>value</i>	The address of variable to hold the value associated with the key.
<i>def</i>	The default value to return in the absence of the key existing in the dictionary.

**Returns**

A Max error code.

**See also**

[dictionary\\_getstring\(\)](#)

**38.8.4.36 dictionary\_getdefsym()**

```
t_max_err dictionary_getdefsym (
    const t_dictionary * d,
    t_symbol * key,
    t_symbol ** value,
    t_symbol * def )
```

Retrieve a `t_symbol*` from the dictionary.

If the named key doesn't exist, then return a specified default value.

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>value</i>	The address of variable to hold the value associated with the key.
<i>def</i>	The default value to return in the absence of the key existing in the dictionary.

**Returns**

A Max error code.

**See also**

[dictionary\\_getsym\(\)](#)

**38.8.4.37 dictionary\_getdictionary()**

```
t_max_err dictionary_getdictionary (
    C74_CONST t_dictionary * d,
    t_symbol * key,
    t_object ** value )
```

Retrieve a [t\\_dictionary](#) pointer from the dictionary.

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>value</i>	The address of variable to hold the value associated with the key.

**Returns**

A Max error code.

**38.8.4.38 dictionary\_getentrycount()**

```
t_atom_long dictionary_getentrycount (
    C74_CONST t_dictionary * d )
```

Return the number of keys in a dictionary.

**Parameters**

<i>d</i>	The dictionary instance.
----------	--------------------------

**Returns**

The number of keys in the dictionary.

**38.8.4.39 dictionary\_getfloat()**

```
t_max_err dictionary_getfloat (
    C74_CONST t_dictionary * d,
    t_symbol * key,
    double * value )
```

Retrieve a double-precision float from the dictionary.

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>value</i>	The address of variable to hold the value associated with the key.

**Returns**

A Max error code.

**38.8.4.40 dictionary\_getkeys()**

```
t_max_err dictionary_getkeys (
    C74_CONST t_dictionary * d,
    long * numkeys,
    t_symbol *** keys )
```

Retrieve all of the key names stored in a dictionary.

The numkeys and keys parameters should be initialized to zero. The [dictionary\\_getkeys\(\)](#) method will allocate memory for the keys it returns. You are then responsible for freeing this memory using [dictionary\\_freekeys\(\)](#). You must use [dictionary\\_freekeys\(\)](#), not some other method for freeing the memory.

**Parameters**

<i>d</i>	The dictionary instance.
<i>numkeys</i>	The address of a long where the number of keys retrieved will be set.
<i>keys</i>	The address of the first of an array <a href="#">t_symbol</a> pointers where the retrieved keys will be set.

**Returns**

A max error code.

**Remarks**

The following example demonstrates fetching all of the keys from a dictionary named 'd' in order to iterate through each item stored in the dictionary.

```
t_symbol **keys = NULL;
long numkeys = 0;
long i;
t_object *anitem;
dictionary_getkeys(d, &numkeys, &keys);
for(i=0; i<numkeys; i++){
    // do something with the keys...
}
if(keys)
    dictionary_freekeys(d, numkeys, keys);
```

**See also**

[dictionary\\_freekeys\(\)](#)

**38.8.4.41 dictionary\_getlong()**

```
t_max_err dictionary_getlong (
    C74_CONST t_dictionary * d,
    t_symbol * key,
    t_atom_long * value )
```

Retrieve a long integer from the dictionary.

**Parameters**

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>value</i>	The address of variable to hold the value associated with the key.

**Returns**

A Max error code.

### 38.8.4.42 dictionary\_getobject()

```
t_max_err dictionary_getobject (
    C74_CONST t_dictionary * d,
    t_symbol * key,
    t_object ** value )
```

Retrieve a [t\\_object](#) pointer from the dictionary.

#### Parameters

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>value</i>	The address of variable to hold the value associated with the key.

#### Returns

A Max error code.

### 38.8.4.43 dictionary\_getstring()

```
t_max_err dictionary_getstring (
    C74_CONST t_dictionary * d,
    t_symbol * key,
    const char ** value )
```

Retrieve a C-string pointer from the dictionary.

The retrieved pointer references the string in the dictionary, it is not a copy.

#### Parameters

<i>d</i>	The dictionary instance.
<i>key</i>	The key associated with the value to lookup.
<i>value</i>	The address of variable to hold the value associated with the key.

#### Returns

A Max error code.

### 38.8.4.44 dictionary\_getsym()

```
t_max_err dictionary_getsym (
    C74_CONST t_dictionary * d,
    t_symbol * key,
    t_symbol ** value )
```

Retrieve a `t_symbol`\* from the dictionary.

#### Parameters

<code>d</code>	The dictionary instance.
<code>key</code>	The key associated with the value to lookup.
<code>value</code>	The address of variable to hold the value associated with the key.

#### Returns

A Max error code.

### 38.8.4.45 dictionary\_hasentry()

```
long dictionary_hasentry (
    C74_CONST t_dictionary * d,
    t_symbol * key )
```

Test a key to set if it exists in the dictionary.

#### Parameters

<code>d</code>	The dictionary instance.
<code>key</code>	The key associated with the value to test.

#### Returns

Returns true if the key exists, otherwise returns false.

### 38.8.4.46 dictionary\_new()

```
t_dictionary* dictionary_new (
    void )
```

Create a new dictionary object.

You can free the dictionary by calling `object_free()`. However, you should keep in mind the guidelines provided in [When to Free a Dictionary](#).

**Returns**

Pointer to the new dictionary object.

**See also**

[object\\_free\(\)](#)

**38.8.4.47 dictionary\_read()**

```
t_max_err dictionary_read (
    const char * filename,
    short path,
    t_dictionary ** d )
```

Read the specified JSON file and return a [t\\_dictionary](#) object.

You are responsible for freeing the dictionary with [object\\_free\(\)](#), subject to the caveats explained in [When to Free a Dictionary](#).

**Parameters**

<i>filename</i>	The name of the file.
<i>path</i>	The path of the file.
<i>d</i>	The address of a <a href="#">t_dictionary</a> pointer that will be set to the newly created dictionary.

**Returns**

A Max error code

**38.8.4.48 dictionary\_read\_yaml()**

```
t_max_err dictionary_read_yaml (
    const char * filename,
    const short path,
    t_dictionary ** d )
```

Read the specified YAML file and return a [t\\_dictionary](#) object.

You are responsible for freeing the dictionary with [object\\_free\(\)](#), subject to the caveats explained in [When to Free a Dictionary](#).

**Parameters**

<i>filename</i>	The name of the file.
<i>path</i>	The path of the file.
<i>d</i>	The address of a <a href="#">t_dictionary</a> pointer that will be set to the newly created dictionary.

**Returns**

A Max error code

**38.8.4.49 dictionary\_sprintf()**

```
t_dictionary* dictionary_sprintf (
    C74_CONST char * fmt,
    ...
)
```

Create a new dictionary populated with values using a combination of attribute and sprintf syntax.

**Parameters**

<i>fmt</i>	An sprintf-style format string specifying key-value pairs with attribute nomenclature.
...	One or more arguments which are to be substituted into the format string.

**Returns**

A new dictionary instance.

**Remarks**

Max attribute syntax is used to define key-value pairs. For example,  
`"@key1 value @key2 another_value"`

One common use of this to create dictionary that represents an element of a patcher, or even an entire patcher itself. The example below creates a dictionary that can be passed to a function like [newobject\\_fromdictionary\(\)](#) to create a new object.

```
t_dictionary *d;
char text[4];
strncpy_zero(text, "foo", 4);
d = dictionary_sprintf("@maxclass comment @varname _name \
    @text \"%s\" @patching_rect %.2f %.2f %.2f %.2f \
    @fontsize %f @textcolor %f %f %f 1.0 \
    @fontname %s @bgcolor 0.001 0.001 0.001 0.",
    text, 20.0, 20.0, 200.0, 24.0,
    18, 0.9, 0.9, 0.9, "Arial");
// do something with the dictionary here.
object_free(d);
```

**See also**

[newobject\\_sprintf\(\)](#)  
[newobject\\_fromdictionary\(\)](#)  
[atom\\_setparse\(\)](#)

### 38.8.4.50 dictionary\_transaction\_lock()

```
long dictionary_transaction_lock (
    t_dictionary * d )
```

Take a lock on a dictionary for preventing dictionary lock for transactions across multiple calls, or holding on to internal dictionary element pointers for complex operations.

#### Parameters

<code>d</code>	The dictionary to lock
----------------	------------------------

#### Returns

A Max error code.

#### See also

[dictionary\\_transaction\\_unlock\(\)](#)

### 38.8.4.51 dictionary\_transaction\_unlock()

```
long dictionary_transaction_unlock (
    t_dictionary * d )
```

Release a lock on a dictionary for preventing dictionary lock for transactions across multiple calls, or holding on to internal dictionary element pointers for complex operations.

#### Parameters

<code>d</code>	The dictionary to unlock
----------------	--------------------------

#### Returns

A Max error code.

#### See also

[dictionary\\_transaction\\_lock\(\)](#)

### 38.8.4.52 dictionary\_write()

```
t_max_err dictionary_write (
    t_dictionary * d,
    const char * filename,
    short path )
```

Serialize the specified [t\\_dictionary](#) object to a JSON file.

#### Parameters

<i>d</i>	The dictionary to serialize into JSON format and write to disk.
<i>filename</i>	The name of the file to write.
<i>path</i>	The path to which the file should be written.

#### Returns

A Max error code.

### 38.8.4.53 dictionary\_write\_yaml()

```
t_max_err dictionary_write_yaml (
    const t_dictionary * d,
    const char * filename,
    const short path )
```

Serialize the specified [t\\_dictionary](#) object to a YAML file.

#### Parameters

<i>d</i>	The dictionary to serialize into YAML format and write to disk.
<i>filename</i>	The name of the file to write.
<i>path</i>	The path to which the file should be written.

#### Returns

A Max error code.

### 38.8.4.54 postdictionary()

```
void postdictionary (
    t_object * d )
```

Print the contents of a dictionary to the Max window.

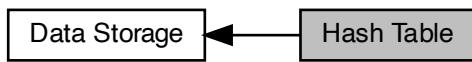
### Parameters

<i>d</i>	A pointer to a dictionary object.
----------	-----------------------------------

## 38.9 Hash Table

A hash table is a data structure that associates some data with a unique key.

Collaboration diagram for Hash Table:



### Data Structures

- struct `t_hashtab_entry`

*A hashtab entry.*

- struct `t_hashtab`

*The hashtab object.*

### Enumerations

- enum

*Default number of slots in the hash table.*

### Functions

- `BEGIN_USING_C_LINKAGE t_hashtab * hashtab_new (long slotcount)`

*Create a new hashtab object.*

- `t_max_err hashtab_store (t_hashtab *x, t_symbol *key, t_object *val)`

*Store an item in a hashtab with an associated key.*

- `t_max_err hashtab_storelong (t_hashtab *x, t_symbol *key, t_atom_long val)`

*Store a `t_atom_long` value in a hashtab with an associated key.*

- `t_max_err hashtab_storesym (t_hashtab *x, t_symbol *key, t_symbol *val)`

*Store a `t_symbol` value in a hashtab with an associated key.*

- `t_max_err hashtab_store_safe (t_hashtab *x, t_symbol *key, t_object *val)`

*Store an item in a hashtab with an associated key.*

- `t_max_err hashtab_storeflags (t_hashtab *x, t_symbol *key, t_object *val, long flags)`  
*Store an item in a hashtab with an associated key and also flags that define the behavior of the item.*
- `t_max_err hashtab_lookup (t_hashtab *x, t_symbol *key, t_object **val)`  
*Return an item stored in a hashtab with the specified key.*
- `t_max_err hashtab_lookuplong (t_hashtab *x, t_symbol *key, t_atom_long *val)`  
*Return a `t_atom_long` value stored in a hashtab with the specified key.*
- `t_max_err hashtab_lookupsym (t_hashtab *x, t_symbol *key, t_symbol **val)`  
*Return a `t_symbol` value stored in a hashtab with the specified key.*
- `t_max_err hashtab_lookupflags (t_hashtab *x, t_symbol *key, t_object **val, long *flags)`  
*Return an item stored in a hashtab with the specified key, also returning the items flags.*
- `t_max_err hashtab_delete (t_hashtab *x, t_symbol *key)`  
*Remove an item from a hashtab associated with the specified key and free it.*
- `t_max_err hashtab_clear (t_hashtab *x)`  
*Delete all items stored in a hashtab.*
- `t_max_err hashtab_chuckkey (t_hashtab *x, t_symbol *key)`  
*Remove an item from a hashtab associated with a given key.*
- `t_max_err hashtab_chuck (t_hashtab *x)`  
*Free a hashtab, but don't free the items it contains.*
- `t_max_err hashtab_findfirst (t_hashtab *x, void **o, long cmpfn(void *, void *), void *cmpdata)`  
*Search the hash table for the first item meeting defined criteria.*
- `t_max_err hashtab_methodall (t_hashtab *x, t_symbol *s,...)`  
*Call the named message on every object in the hashtab.*
- `t_max_err hashtab_funall (t_hashtab *x, method fun, void *arg)`  
*Call the specified function for every item in the hashtab.*
- `t_atom_long hashtab_getsize (t_hashtab *x)`  
*Return the number of items stored in a hashtab.*
- `void hashtab_print (t_hashtab *x)`  
*Post a hashtab's statistics to the Max window.*
- `void hashtab_READONLY (t_hashtab *x, long readonly)`  
*Set the hashtab's readonly bit.*
- `void hashtab_FLAGS (t_hashtab *x, long flags)`  
*Set the hashtab's datastore flags.*
- `t_atom_long hashtab_getflags (t_hashtab *x)`  
*Get the hashtab's datastore flags.*
- `t_max_err hashtab_KEYFLAGS (t_hashtab *x, t_symbol *key, long flags)`  
*Change the flags for an item stored in the hashtab with a given key.*
- `t_atom_long hashtab_GETKEYFLAGS (t_hashtab *x, t_symbol *key)`  
*Retrieve the flags for an item stored in the hashtab with a given key.*
- `t_max_err hashtab_getkeys (t_hashtab *x, long *kc, t_symbol ***kv)`  
*Retrieve all of the keys stored in a hashtab.*

### 38.9.1 Detailed Description

A hash table is a data structure that associates some data with a unique key.

If you know the key, you can get back the data much more quickly than with a linked list, particularly as the number of items stored grows larger. The Max hash table `t_hashtab` is optimized to work with symbol pointers as keys, but you can use any pointer or number, as long as it is unique.

To create a `t_hashtab`, you use `hashtab_new()`. To add items, use `hashtab_store()`. To find items that have been added, use `hashtab_lookup()`.

By contrast with linked lists and arrays, hash tables do not have a strong sense of ordering. You can iterate through all items using `hashtab_funall()`, but the exact order is not under your control as items are added and removed. There is also no way to "sort" a hash table.

Example:

The following example creates a hashtab, shows how to add some data (in this case, just a number), look it up, and delete the hashtab.

```
t_hashtab *tab = (t_hashtab *)hashtab_new(0);
long result, value;
hashtab_store(tab, gensym("a great number"), (t_object *)74);
result = hashtab_lookup(tab, gensym("a great number"), (t_object **)value);
if (!result)
    post("found the value and it is %ld", value);
else
    post("did not find the value");
hashtab_chuck(tab);
```

Note that the Max `t_dictionary` used for managing patcher data is implemented internally using both a `t_hashtab` and a `t_linklist` in parallel. The `t_hashtab` provides fast access, and the `t_linklist` provides sorting.

See also

[http://en.wikipedia.org/wiki/Hash\\_table](http://en.wikipedia.org/wiki/Hash_table)

[Linked List](#)

### 38.9.2 Enumeration Type Documentation

#### 38.9.2.1 anonymous enum

anonymous enum

Default number of slots in the hash table.

Creating a hashtab using `hashtab_new()` with an argument of 0 will use the default number of slots. Primes typically work well for the number of slots.

### 38.9.3 Function Documentation

#### 38.9.3.1 hashtab\_chuck()

```
t_max_err hashtab_chuck (
    t_hashtab * x )
```

Free a hashtab, but don't free the items it contains.

The hashtab can contain a variety of different types of data. By default, the hashtab assumes that all items are max objects with a valid [t\\_object](#) header.

You can alter the hashtab's notion of what it contains by using the [hashtab\\_flags\(\)](#) method.

When you free the hashtab by calling [object\\_free\(\)](#) it then tries to free all of the items it contains. If the hashtab is storing a custom type of data, or should otherwise not free the data it contains, then call [hashtab\\_chuck\(\)](#) to free the object instead of [object\\_free\(\)](#).

##### Parameters

x	The hashtab object to be freed.
---	---------------------------------

##### Returns

A max error code.

##### See also

[object\\_free](#)

#### 38.9.3.2 hashtab\_chuckkey()

```
t_max_err hashtab_chuckkey (
    t_hashtab * x,
    t_symbol * key )
```

Remove an item from a hashtab associated with a given key.

You are responsible for freeing any memory associated with the item that is removed from the hashtab.

##### Parameters

x	The hashtab instance.
key	The key of the item to delete.

**Returns**

A Max error code.

**See also**

[hashtab\\_delete](#)

### 38.9.3.3 hashtab\_clear()

```
t_max_err hashtab_clear (
    t_hashtab * x )
```

Delete all items stored in a hashtab.

This is the equivalent of calling [hashtab\\_delete\(\)](#) on every item in a hashtab.

**Returns**

A max error code.

**See also**

[hashtab\\_flags\(\)](#)  
[hashtab\\_delete\(\)](#)

### 38.9.3.4 hashtab\_delete()

```
t_max_err hashtab_delete (
    t_hashtab * x,
    t_symbol * key )
```

Remove an item from a hashtab associated with the specified key and free it.

The hashtab can contain a variety of different types of data. By default, the hashtab assumes that all items are max objects with a valid [t\\_object](#) header. Thus by default, it frees items by calling [object\\_free\(\)](#) on them.

You can alter the hashtab's notion of what it contains by using the [hashtab\\_flags\(\)](#) method.

If you wish to remove an item from the hashtab and free it yourself, then you should use [hashtab\\_chuckkey\(\)](#).

**Parameters**

<i>x</i>	The hashtab instance.
<i>key</i>	The key of the item to delete.

**Returns**

A Max error code.

**See also**

[hashtab\\_chuckkey\(\)](#)  
[hashtab\\_clear\(\)](#)  
[hashtab\\_flags\(\)](#)

**38.9.3.5 hashtab\_findfirst()**

```
t_max_err hashtab_findfirst (
    t_hashtab * x,
    void ** o,
    long cmpfn(void *, void *,
    void * cmpdata ))
```

Search the hash table for the first item meeting defined criteria.

The items in the hashtab are iteratively processed, calling a specified comparison function on each until the comparison function returns true.

**Parameters**

<i>x</i>	The hashtab instance.
<i>o</i>	The address to pointer that will be set with the matching item.
<i>cmpfn</i>	The function used to determine a match in the list.
<i>cmpdata</i>	An argument to be passed to the <a href="#">t_cmpfn</a> . This will be passed as the second of the two args to the <a href="#">t_cmpfn</a> . The first arg will be the hashtab item at each iteration in the list.

**Returns**

A max error code.

**See also**

[linklist\\_findfirst\(\)](#)  
[t\\_cmpfn](#)

### 38.9.3.6 hashtable\_flags()

```
void hashtable_flags (
    t_hashtable * x,
    long flags )
```

Set the hashtable's datastore flags.

The available flags are enumerated in [e\\_max\\_datastore\\_flags](#). These flags control the behavior of the hashtable, particularly when removing items from the list using functions such as [hashtable\\_clear\(\)](#), [hashtable\\_delete\(\)](#), or when freeing the hashtable itself.

#### Parameters

<i>x</i>	The hashtable instance.
<i>flags</i>	A valid value from the <a href="#">e_max_datastore_flags</a> . The default is <a href="#">OBJ_FLAG_OBJ</a> .

### 38.9.3.7 hashtable\_funall()

```
t_max_err hashtable_funall (
    t_hashtable * x,
    method fun,
    void * arg )
```

Call the specified function for every item in the hashtable.

#### Parameters

<i>x</i>	The hashtable instance.
<i>fun</i>	The function to call, specified as function pointer cast to a Max #method.
<i>arg</i>	An argument that you would like to pass to the function being called.

#### Returns

A max error code.

#### Remarks

The [hashtable\\_funall\(\)](#) method will call your function for every item in the list. It will pass both a pointer to the item in the list, and any argument that you provide. The following example shows a function that could be called by [hashtable\\_funall\(\)](#).

```
void myFun(t_hashtable_entry *e, void *myArg)
{
    if (e->key && e->value) {
        // do something with e->key, e->value, and myArg here as appropriate
    }
}
```

Referenced by [jit\\_class\\_addinterface\(\)](#).

### 38.9.3.8 hashtab\_getflags()

```
t_atom_long hashtab_getflags (
    t_hashtab * x )
```

Get the hashtab's datastore flags.

#### Parameters

x	The hashtab instance.
---	-----------------------

#### Returns

The current state of the hashtab flags as enumerated in [e\\_max\\_datastore\\_flags](#).

### 38.9.3.9 hashtab\_getkeyflags()

```
t_atom_long hashtab_getkeyflags (
    t_hashtab * x,
    t_symbol * key )
```

Retrieve the flags for an item stored in the hashtab with a given key.

#### Parameters

x	The hashtab instance.
key	The key in the hashtab whose flags will be returned.

#### Returns

The flags for the given key.

#### See also

[hashtab\\_store\\_flags\(\)](#)

### 38.9.3.10 hashtab\_getkeys()

```
t_max_err hashtab_getkeys (
    t_hashtab * x,
```

```
long * kc,
t_symbol *** kv )
```

Retrieve all of the keys stored in a hashtab.

If the kc and kv parameters are properly initialized to zero, then [hashtab\\_getkeys\(\)](#) will allocate memory for the keys it returns. You are then responsible for freeing this memory using [sysmem\\_freeptr\(\)](#).

#### Parameters

x	The hashtab instance.
kc	The address of a long where the number of keys retrieved will be set.
kv	The address of the first of an array <a href="#">t_symbol</a> pointers where the retrieved keys will be set.

#### Returns

A max error code.

#### Remarks

The following example demonstrates fetching all of the keys from a hashtab in order to iterate through each item stored in the hashtab.

```
t_symbol **keys = NULL;
long numKeys = 0;
long i;
t_object *anItem;
hashtab_getkeys(aHashtab, &numKeys, &keys);
for(i=0; i<numKeys; i++){
    hashtab_lookup(aHashtab, keys[i], &anItem);
    // Do something with anItem here...
}
if(keys)
    sysmem_freeptr(keys);
```

### 38.9.3.11 hashtab\_getsize()

```
t_atom_long hashtab_getsize (
    t_hashtab * x )
```

Return the number of items stored in a hashtab.

#### Parameters

x	The hashtab instance.
---	-----------------------

#### Returns

The number of items in the hash table.

### 38.9.3.12 hashtable\_keyflags()

```
t_max_err hashtable_keyflags (
    t_hashtable * x,
    t_symbol * key,
    long flags )
```

Change the flags for an item stored in the hashtable with a given key.

#### Parameters

<i>x</i>	The hashtable instance.
<i>key</i>	The key in the hashtable whose flags will be changed.
<i>flags</i>	One of the values listed in <a href="#">e_max_datastore_flags</a> .

#### Returns

A Max error code.

#### See also

[hashtable\\_store\\_flags\(\)](#)

### 38.9.3.13 hashtable\_lookup()

```
t_max_err hashtable_lookup (
    t_hashtable * x,
    t_symbol * key,
    t_object ** val )
```

Return an item stored in a hashtable with the specified key.

#### Parameters

<i>x</i>	The hashtable instance.
<i>key</i>	The key in the hashtable to fetch.
<i>val</i>	The address of a pointer to which the fetched value will be assigned.

#### Returns

A Max error code.

## See also

[hashtab\\_store\(\)](#), [hashtab\\_lookuplong\(\)](#), [hashtab\\_lookupsym\(\)](#)

**38.9.3.14 hashtab\_lookupflags()**

```
t_max_err hashtab_lookupflags (
    t_hashtab * x,
    t_symbol * key,
    t_object ** val,
    long * flags )
```

Return an item stored in a hashtab with the specified key, also returning the items flags.

## Parameters

<i>x</i>	The hashtab instance.
<i>key</i>	The key in the hashtab to fetch.
<i>val</i>	The address of a pointer to which the fetched value will be assigned.
<i>flags</i>	The address of a value to which the fetched flags will be assigned.

## Returns

A Max error code.

## See also

[hashtab\\_lookup\(\)](#)  
[hashtab\\_store\\_flags\(\)](#)

**38.9.3.15 hashtab\_lookuplong()**

```
t_max_err hashtab_lookuplong (
    t_hashtab * x,
    t_symbol * key,
    t_atom_long * val )
```

Return a *t\_atom\_long* value stored in a hashtab with the specified key.

## Parameters

<i>x</i>	The hashtab instance.
<i>key</i>	The key in the hashtab to fetch.
<i>val</i>	A pointer to a <i>t_atom_long</i> to which the fetched value will be assigned.

**Returns**

A Max error code.

**See also**

[hashtab\\_storelong\(\)](#), [hashtab\\_lookup\(\)](#), [hashtab\\_lookupsym\(\)](#)

**38.9.3.16 hashtab\_lookupsym()**

```
t_max_err hashtab_lookupsym (
    t_hashtab * x,
    t_symbol * key,
    t_symbol ** val )
```

Return a [t\\_symbol](#) value stored in a hashtab with the specified key.

**Parameters**

<i>x</i>	The hashtab instance.
<i>key</i>	The key in the hashtab to fetch.
<i>val</i>	A pointer to the address of a <a href="#">t_symbol</a> to which the fetched value will be assigned.

**Returns**

A Max error code.

**See also**

[hashtab\\_storesym\(\)](#), [hashtab\\_lookup\(\)](#), [hashtab\\_lookuplong\(\)](#)

**38.9.3.17 hashtab\_methodall()**

```
t_max_err hashtab_methodall (
    t_hashtab * x,
    t_symbol * s,
    ... )
```

Call the named message on every object in the hashtab.

The [hashtab\\_methodall\(\)](#) function requires that all items in the hashtab are object instances with a valid [t\\_object](#) header.

**Parameters**

<i>x</i>	The hashtab instance.
<i>s</i>	The name of the message to send to the objects.
...	Any arguments to be sent with the message.

**Returns**

A max error code.

**Remarks**

Internally, this function uses [object\\_method\(\)](#), meaning that no errors will be posted if the message name does not exist for the object. It also means that messages sent methods with [A\\_GIMME](#) definitions will need to be given a symbol argument prior to the argc and argv array information.

**38.9.3.18 hashtab\_new()**

```
BEGIN_USING_C_LINKAGE t_hashtab* hashtab_new (
    long slotcount )
```

Create a new hashtab object.

You can free the hashtab by calling [object\\_free\(\)](#) on the hashtab's pointer, or by using [hashtab\\_chuck\(\)](#).

**Parameters**

<i>slotcount</i>	The number of slots in the hash table. Prime numbers typically work well. Pass 0 to get the default size.
------------------	---

**Returns**

Pointer to the new hashtab object.

**See also**

[HASH\\_DEFSLOTS](#)  
[object\\_free\(\)](#)  
[hashtab\\_chuck\(\)](#)

Referenced by [jit\\_class\\_addinterface\(\)](#).

### 38.9.3.19 hashtable\_print()

```
void hashtable_print (
    t_hashtable * x )
```

Post a hashtable's statistics to the Max window.

#### Parameters

<i>x</i>	The hashtable instance.
----------	-------------------------

### 38.9.3.20 hashtable\_READONLY()

```
void hashtable_READONLY (
    t_hashtable * x,
    long readonly )
```

Set the hashtable's readonly bit.

By default the readonly bit is 0, indicating that it is threadsafe for both reading and writing. Setting the readonly bit to 1 will disable the hashtable's threadsafety mechanism, increasing performance but at the expense of threadsafe operation. Unless you can guarantee the threading context for a hashtable's use, you should leave this set to 0.

#### Parameters

<i>x</i>	The hashtable instance.
<i>readonly</i>	A 1 or 0 for setting the readonly bit.

### 38.9.3.21 hashtable\_store()

```
t_max_err hashtable_store (
    t_hashtable * x,
    t_symbol * key,
    t_object * val )
```

Store an item in a hashtable with an associated key.

#### Parameters

<i>x</i>	The hashtable instance.
<i>key</i>	The key in the hashtable with which to associate the value.
<i>val</i>	The value to store.

**Returns**

A Max error code.

**See also**

[hashtab\\_lookup\(\)](#), [hashtab\\_storesafe\(\)](#), [hashtab\\_storelong\(\)](#), [hashtab\\_storesym\(\)](#)

Referenced by `jit_class_addinterface()`.

**38.9.3.22 hashtab\_store\_safe()**

```
t_max_err hashtab_store_safe (
    t_hashtab * x,
    t_symbol * key,
    t_object * val )
```

Store an item in a hashtab with an associated key.

The difference between `hashtab_store_safe()` and `hashtab_store()` is what happens in the event of a collision in the hash table. The normal `hashtab_store()` function will free the existing value at the collision location with `sysmem_freeptr()` and then replaces it. This version doesn't try to free the existing value at the collision location, but instead just over-writes it.

**Parameters**

<code>x</code>	The hashtab instance.
<code>key</code>	The key in the hashtab with which to associate the value.
<code>val</code>	The value to store.

**Returns**

A Max error code.

**See also**

[hashtab\\_store\(\)](#)

**38.9.3.23 hashtab\_storeflags()**

```
t_max_err hashtab_storeflags (
    t_hashtab * x,
    t_symbol * key,
    t_object * val,
    long flags )
```

Store an item in a hashtab with an associated key and also flags that define the behavior of the item.

The `hashtab_store()` method is the same as calling this method with the default (0) flags.

**Parameters**

<i>x</i>	The hashtab instance.
<i>key</i>	The key in the hashtab with which to associate the value.
<i>val</i>	The value to store.
<i>flags</i>	One of the values listed in <a href="#">e_max_datastore_flags</a> .

**Returns**

A Max error code.

**See also**

[hashtab\\_store\(\)](#)

**38.9.3.24 hashtab\_storelong()**

```
t_max_err hashtab_storelong (
    t_hashtab * x,
    t_symbol * key,
    t_atom_long val )
```

Store a `t_atom_long` value in a hashtab with an associated key.

**Parameters**

<i>x</i>	The hashtab instance.
<i>key</i>	The key in the hashtab with which to associate the value.
<i>val</i>	The <code>t_atom_long</code> value to store.

**Returns**

A Max error code.

**See also**

[hashtab\\_lookuplong\(\)](#), [hashtab\\_store\(\)](#), [hashtab\\_storesafe\(\)](#), [hashtab\\_storesym\(\)](#)

### 38.9.3.25 hashtab\_storesym()

```
t_max_err hashtab_storesym (
    t_hashtab * x,
    t_symbol * key,
    t_symbol * val )
```

Store a `t_symbol` value in a hashtab with an associated key.

#### Parameters

<code>x</code>	The hashtab instance.
<code>key</code>	The key in the hashtab with which to associate the value.
<code>val</code>	The <code>t_symbol</code> pointer to store.

#### Returns

A Max error code.

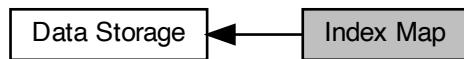
#### See also

[hashtab\\_lookupsym\(\)](#), [hashtab\\_store\(\)](#), [hashtab\\_storesafe\(\)](#), [hashtab\\_storelong\(\)](#)

## 38.10 Index Map

An indexmap is basically a managed array of pointers, but it allows you to derive relatively quickly the index from a pointer in the array.

Collaboration diagram for Index Map:



## Data Structures

- struct `t_indexmap_entry`  
*An indexmap element.*
- struct `t_indexmap`  
*An indexmap object.*

## Functions

- `t_indexmap * indexmap_new (void)`  
*Create a new indexmap object.*
- `void indexmap_append (t_indexmap *x, void *data)`  
*Add an item to an indexmap.*
- `t_max_err indexmap_move (t_indexmap *x, void *data, long newindex)`  
*Move an item to a different position in an indexmap.*
- `t_max_err indexmap_delete (t_indexmap *x, void *data)`  
*Delete a specified item from an indexmap.*
- `t_max_err indexmap_delete_index (t_indexmap *x, long index)`  
*Delete an item from the indexmap by index.*
- `t_max_err indexmap_delete_multi (t_indexmap *x, long count, void **pdata)`  
*Delete multiple specified items from an indexmap.*
- `t_max_err indexmap_delete_index_multi (t_indexmap *x, long count, long *indices)`  
*Delete multiple items from an indexmap by index.*
- `void * indexmap_datafromindex (t_indexmap *x, long index)`  
*Get an item from an indexmap by index.*
- `t_max_err indexmap_indexfromdata (t_indexmap *x, void *data, long *index)`  
*Find the index of an item given a pointer to the item.*
- `long indexmap_getsize (t_indexmap *x)`  
*Return the number of items in an indexmap.*
- `void indexmap_clear (t_indexmap *x)`  
*Delete all items in an indexmap.*
- `void indexmap_sort (t_indexmap *x, t_cmpfn fn)`  
*Sort the items in an indexmap.*

### 38.10.1 Detailed Description

An indexmap is basically a managed array of pointers, but it allows you to derive relatively quickly the index from a pointer in the array.

The index is assumed to be 0-N (where N is the current size of the array). You can sort the data and retain access to an index from the data relatively quickly. There is a hashtab which holds pieces of memory that hold indices that can be referenced by the data pointer. There is also an array of data pointers – this is in "index" order. When operations take place on the array (insert, delete, sort), the pointers in the hashtab are updated with new indices.

### 38.10.2 Function Documentation

#### 38.10.2.1 `indexmap_append()`

```
void indexmap_append (
    t_indexmap * x,
    void * data )
```

Add an item to an indexmap.

**Parameters**

<i>x</i>	The indexmap instance.
<i>data</i>	The item to add.

**38.10.2.2 indexmap\_clear()**

```
void indexmap_clear (
    t_indexmap * x )
```

Delete all items in an indexmap.

**Parameters**

<i>x</i>	The indexmap instance.
----------	------------------------

**38.10.2.3 indexmap\_datafromindex()**

```
void* indexmap_datafromindex (
    t_indexmap * x,
    long index )
```

Get an item from an indexmap by index.

**Parameters**

<i>x</i>	The indexmap instance.
<i>index</i>	The index from which to fetch a stored item.

**Returns**

The item stored at the specified index.

**38.10.2.4 indexmap\_delete()**

```
t_max_err indexmap_delete (
    t_indexmap * x,
    void * data )
```

Delete a specified item from an indexmap.

**Parameters**

<i>x</i>	The indexmap instance.
<i>data</i>	The item pointer to remove from the indexmap.

**Returns**

A Max error code.

**38.10.2.5 indexmap\_delete\_index()**

```
t_max_err indexmap_delete_index (
    t_indexmap * x,
    long index )
```

Delete an item from the indexmap by index.

**Parameters**

<i>x</i>	The indexmap instance.
<i>index</i>	The index of the item to remove from the indexmap.

**Returns**

A Max error code.

**38.10.2.6 indexmap\_delete\_index\_multi()**

```
t_max_err indexmap_delete_index_multi (
    t_indexmap * x,
    long count,
    long * indices )
```

Delete multiple items from an indexmap by index.

**Parameters**

<i>x</i>	The indexmap instance.
<i>count</i>	The number of items to remove from the indexmap.
<i>indices</i>	The address of the first of an array of index numbers to remove the indexmap.

**Returns**

A Max error code.

**38.10.2.7 indexmap\_delete\_multi()**

```
t_max_err indexmap_delete_multi (
    t_indexmap * x,
    long count,
    void ** pdata )
```

Delete multiple specified items from an indexmap.

**Parameters**

<i>x</i>	The indexmap instance.
<i>count</i>	The number of items to remove from the indexmap.
<i>pdata</i>	The address of the first of an array of item pointers to remove from the indexmap.

**Returns**

A Max error code.

**38.10.2.8 indexmap\_getsize()**

```
long indexmap_getsize (
    t_indexmap * x )
```

Return the number of items in an indexmap.

**Parameters**

<i>x</i>	The indexmap instance.
----------	------------------------

**Returns**

The number of items in the indexmap.

### 38.10.2.9 indexmap\_indexfromdata()

```
t_max_err indexmap_indexfromdata (
    t_indexmap * x,
    void * data,
    long * index )
```

Find the index of an item given a pointer to the item.

#### Parameters

<i>x</i>	The indexmap instance.
<i>data</i>	The item whose index you wish to look up.
<i>index</i>	The address of a variable to hold the retrieved index.

#### Returns

A Max error code.

### 38.10.2.10 indexmap\_move()

```
t_max_err indexmap_move (
    t_indexmap * x,
    void * data,
    long newindex )
```

Move an item to a different position in an indexmap.

#### Parameters

<i>x</i>	The indexmap instance.
<i>data</i>	The item in the indexmap to move.
<i>newindex</i>	The new index to which to move the item.

#### Returns

A Max error code.

### 38.10.2.11 indexmap\_new()

```
t_indexmap* indexmap_new (
    void )
```

Create a new indexmap object.

**Returns**

Pointer to the new indexmap object.

**38.10.2.12 indexmap\_sort()**

```
void indexmap_sort (
    t_indexmap * x,
    t_cmpfn fn )
```

Sort the items in an indexmap.

Items are sorted using a [t\\_cmpfn](#) function that is passed in as an argument.

**Parameters**

<i>x</i>	The indexmap instance.
<i>fn</i>	The function used to sort the list.

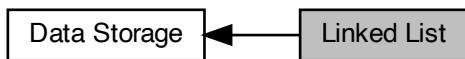
**See also**

[linklist\\_sort\(\)](#)

**38.11 Linked List**

The Max [t\\_linklist](#) data structure is useful for maintaining ordered lists of items where you want to be able to insert and delete items efficiently.

Collaboration diagram for Linked List:

**Data Structures**

- struct [t\\_llitem](#)  
*A linklist element.*
- struct [t\\_linklist](#)  
*The linklist object.*

## Functions

- **BEGIN\_USING\_C\_LINKAGE t\_linklist \* linklist\_new (void)**  
*Create a new linklist object.*
- **void linklist\_chuck (t\_linklist \*x)**  
*Free a linklist, but don't free the items it contains.*
- **t\_atom\_long linklist\_getsize (t\_linklist \*x)**  
*Return the number of items in a linklist object.*
- **void \* linklist\_getindex (t\_linklist \*x, long index)**  
*Return the item stored in a linklist at a specified index.*
- **t\_atom\_long linklist\_objptr2index (t\_linklist \*x, void \*p)**  
*Return an item's index, given the item itself.*
- **t\_atom\_long linklist\_append (t\_linklist \*x, void \*o)**  
*Add an item to the end of the list.*
- **t\_atom\_long linklist\_insertindex (t\_linklist \*x, void \*o, long index)**  
*Insert an item into the list at the specified index.*
- **long linklist\_insert\_sorted (t\_linklist \*x, void \*o, long cmpfn(void \*, void \*))**  
*Insert an item into the list, keeping the list sorted according to a specified comparison function.*
- **t\_llelem \* linklist\_insertafterobjptr (t\_linklist \*x, void \*o, void \*objptr)**  
*Insert an item into the list after another specified item.*
- **t\_llelem \* linklist\_insertbeforeobjptr (t\_linklist \*x, void \*o, void \*objptr)**  
*Insert an item into the list before another specified item.*
- **t\_llelem \* linklist\_moveafterobjptr (t\_linklist \*x, void \*o, void \*objptr)**  
*Move an existing item in the list to a position after another specified item in the list.*
- **t\_llelem \* linklist\_movebeforeobjptr (t\_linklist \*x, void \*o, void \*objptr)**  
*Move an existing item in the list to a position before another specified item in the list.*
- **t\_atom\_long linklist\_deleteindex (t\_linklist \*x, long index)**  
*Remove the item from the list at the specified index and free it.*
- **long linklist\_chuckindex (t\_linklist \*x, long index)**  
*Remove the item from the list at the specified index.*
- **long linklist\_chuckobject (t\_linklist \*x, void \*o)**  
*Remove the specified item from the list.*
- **long linklist\_deleteobject (t\_linklist \*x, void \*o)**  
*Delete the specified item from the list.*
- **void linklist\_clear (t\_linklist \*x)**  
*Remove and free all items in the list.*
- **t\_atom\_long linklist\_makearray (t\_linklist \*x, void \*\*a, long max)**  
*Retrieve linklist items as an array of pointers.*
- **void linklist\_reverse (t\_linklist \*x)**  
*Reverse the order of items in the linked-list.*
- **void linklist\_rotate (t\_linklist \*x, long i)**  
*Rotate items in the linked list in circular fashion.*
- **void linklist\_shuffle (t\_linklist \*x)**  
*Randomize the order of items in the linked-list.*
- **void linklist\_swap (t\_linklist \*x, long a, long b)**  
*Swap the position of two items in the linked-list, specified by index.*
- **t\_atom\_long linklist\_findfirst (t\_linklist \*x, void \*\*o, long cmpfn(void \*, void \*), void \*cmpdata)**

- void [linklist\\_findall](#) ([t\\_linklist](#) \*x, [t\\_linklist](#) \*\*out, long cmpfn(void \*, void \*), void \*cmpdata)
 

*Search the linked list for the first item meeting defined criteria.*
- void [linklist\\_methodall](#) ([t\\_linklist](#) \*x, [t\\_symbol](#) \*s,...)
 

*Search the linked list for all items meeting defined criteria.*
- void \* [linklist\\_methodindex](#) ([t\\_linklist](#) \*x, [t\\_atom\\_long](#) i, [t\\_symbol](#) \*s,...)
 

*Call the named message on every object in the linklist.*
- void [linklist\\_sort](#) ([t\\_linklist](#) \*x, long cmpfn(void \*, void \*))
 

*Call the named message on an object specified by index.*
- void [linklist\\_funall](#) ([t\\_linklist](#) \*x, method fun, void \*arg)
 

*Call the specified function for every item in the linklist.*
- [t\\_atom\\_long](#) [linklist\\_funall\\_break](#) ([t\\_linklist](#) \*x, method fun, void \*arg)
 

*Call the specified function for every item in the linklist.*
- void \* [linklist\\_funindex](#) ([t\\_linklist](#) \*x, long i, method fun, void \*arg)
 

*Call the specified function for an item specified by index.*
- void \* [linklist\\_substitute](#) ([t\\_linklist](#) \*x, void \*p, void \*newp)
 

*Given an item in the list, replace it with a different value.*
- void \* [linklist\\_next](#) ([t\\_linklist](#) \*x, void \*p, void \*\*next)
 

*Given an item in the list, find the next item.*
- void \* [linklist\\_prev](#) ([t\\_linklist](#) \*x, void \*p, void \*\*prev)
 

*Given an item in the list, find the previous item.*
- void \* [linklist\\_last](#) ([t\\_linklist](#) \*x, void \*\*item)
 

*Return the last item (the tail) in the linked-list.*
- void [linklist\\_READONLY](#) ([t\\_linklist](#) \*x, long readonly)
 

*Set the linklist's readonly bit.*
- void [linklist\\_FLAGS](#) ([t\\_linklist](#) \*x, long flags)
 

*Set the linklist's datastore flags.*
- [t\\_atom\\_long](#) [linklist\\_GETFLAGS](#) ([t\\_linklist](#) \*x)
 

*Get the linklist's datastore flags.*
- long [linklist\\_MATCH](#) (void \*a, void \*b)
 

*A linklist comparison method that determines if two item pointers are equal.*

### 38.11.1 Detailed Description

The Max [t\\_linklist](#) data structure is useful for maintaining ordered lists of items where you want to be able to insert and delete items efficiently.

Random access of individual items, however, gets appreciably slower as the list grows in size. The [t\\_linklist](#) is thread-safe by default, but thread safety can be turned off for performance benefits in single-threaded applications. However, ensure that your use of the linked list is truly single-threaded (based on an understanding of Max's [Threading](#) model) before turning off the thread safety features.

By default, the [t\\_linklist](#) holds pointers to Max objects. However, you can treat what the linklist holds as data rather than objects to be freed by using the [linklist\\_FLAGS\(\)](#) function.

Here is a simple example of the use of [t\\_linklist](#). The code below stores five symbols, sorts them, searches for a specific item, deletes an item, prints all items, and then frees the entire structure. Since symbols in Max are never freed, [linklist\\_FLAGS\(\)](#) is used to specify that data, rather than object pointers, are being stored.

```

void mylistfun()
{
    t_linklist *list;
    list = (t_linklist *)linklist_new();
    linklist_flags(list, OBJ_FLAG_DATA);
    // add some data
    linklist_append(list, gensym("one"));
    linklist_append(list, gensym("two"));
    linklist_append(list, gensym("three"));
    linklist_append(list, gensym("four"));
    linklist_append(list, gensym("five"));
    // sort
    linklist_sort(list, (t_cmpfn)mysortfun);
    // search
    index = linklist_findfirst(list, &found, mysearchfun, gensym("four")); // find the "four" symbol
    if (index != -1) // found
        linklist_chuckindex(list, index);
    // iterate
    linklist_funall(list, myprintfun, NULL);
    // delete
    linklist_chuck(list);
}

```

The sorting function compares two items in the list and returns non-zero if the first one should go before the second one.

```

long mysortfun(void *a, void *b)
{
    t_symbol *sa = (t_symbol *)a;
    t_symbol *sb = (t_symbol *)b;
    return strcmp(sa->s_name, sb->s_name) > 0;
}

```

The search function is passed the final argument to `linklist_findfirst()` and, in this case, just returns the symbol that matches, which is just testing for pointer equivalence since all Max symbols are unique. You could do more sophisticated searching if you store more complex data in a linklist.

```

long mysearchfun(t_symbol *elem, t_symbol *match)
{
    return elem == match;
}

```

The iteration function takes some action on all items in the list. The third argument to `linklist_funall()` is passed as the second argument to your iteration function. In this example, we don't do anything with it.

```

void myprintfun(t_symbol *item, void *dummy)
{
    post ("%s", item->s_name);
}

```

#### See also

[http://en.wikipedia.org/wiki/Linked\\_list](http://en.wikipedia.org/wiki/Linked_list)

## 38.11.2 Function Documentation

### 38.11.2.1 linklist\_append()

```

t_atom_long linklist_append (
    t_linklist * x,
    void * o )

```

Add an item to the end of the list.

**Parameters**

<i>x</i>	The linklist instance.
<i>o</i>	The item pointer to append to the linked-list.

**Returns**

The updated size of the linklist after appending the new item, or -1 if the append failed.

Referenced by [jit\\_linklist\\_append\(\)](#).

**38.11.2.2 linklist\_chuck()**

```
void linklist_chuck (
    t_linklist * x )
```

Free a linklist, but don't free the items it contains.

The linklist can contain a variety of different types of data. By default, the linklist assumes that all items are max objects with a valid [t\\_object](#) header.

You can alter the linklist's notion of what it contains by using the [linklist\\_flags\(\)](#) method.

When you free the linklist by calling [object\\_free\(\)](#) it then tries to free all of the items it contains. If the linklist is storing a custom type of data, or should otherwise not free the data it contains, then call [linklist\\_chuck\(\)](#) to free the object instead of [object\\_free\(\)](#).

**Parameters**

<i>x</i>	The linklist object to be freed.
----------	----------------------------------

**See also**

[object\\_free](#)

Referenced by [jit\\_linklist\\_chuck\(\)](#), and [jit\\_linklist\\_findcount\(\)](#).

**38.11.2.3 linklist\_chuckindex()**

```
long linklist_chuckindex (
    t_linklist * x,
    long index )
```

Remove the item from the list at the specified index.

You are responsible for freeing any memory associated with the item that is removed from the linklist.

**Parameters**

<i>x</i>	The linklist instance.
<i>index</i>	The index of the item to remove.

**Returns**

Returns [MAX\\_ERR\\_NONE](#) on successful removal, otherwise returns [MAX\\_ERR\\_GENERIC](#)

**See also**

[linklist\\_deleteindex](#)

[linklist\\_chuckobject](#)

Referenced by [jit\\_linklist\\_chuckindex\(\)](#).

**38.11.2.4 linklist\_chuckobject()**

```
long linklist_chuckobject (
    t_linklist * x,
    void * o )
```

Remove the specified item from the list.

You are responsible for freeing any memory associated with the item that is removed from the linklist.

**Parameters**

<i>x</i>	The linklist instance.
<i>o</i>	The pointer to the item to remove.

**See also**

[linklist\\_deleteindex](#)

[linklist\\_chuckindex](#)

[linklist\\_deleteobject](#)

**38.11.2.5 linklist\_clear()**

```
void linklist_clear (
    t_linklist * x )
```

Remove and free all items in the list.

Freeing items in the list is subject to the same rules as [linklist\\_deleteindex\(\)](#). You can alter the linklist's notion of what it contains, and thus how items are freed, by using the [linklist\\_flags\(\)](#) method.

#### Parameters

<code>x</code>	The linklist instance.
----------------	------------------------

Referenced by [jit\\_linklist\\_clear\(\)](#), and [jit\\_linklist\\_free\(\)](#).

### 38.11.2.6 linklist\_deleteindex()

```
t_atom_long linklist_deleteindex (
    t_linklist * x,
    long index )
```

Remove the item from the list at the specified index and free it.

The linklist can contain a variety of different types of data. By default, the linklist assumes that all items are max objects with a valid [t\\_object](#) header. Thus by default, it frees items by calling [object\\_free\(\)](#) on them.

You can alter the linklist's notion of what it contains by using the [linklist\\_flags\(\)](#) method.

If you wish to remove an item from the linklist and free it yourself, then you should use [linklist\\_chuckptr\(\)](#).

#### Parameters

<code>x</code>	The linklist instance.
<code>index</code>	The index of the item to delete.

#### Returns

Returns the index number of the item deleted, or -1 if the operation failed.

#### See also

[linklist\\_chuckindex](#)  
[linklist\\_chuckobject](#)

Referenced by [jit\\_linklist\\_deleteindex\(\)](#).

### 38.11.2.7 linklist\_deleteobject()

```
long linklist_deleteobject (
    t_linklist * x,
    void * o )
```

Delete the specified item from the list.

The object is removed from the list and deleted. The deletion is done with respect to any flags passed to linklist\_flags.

#### Parameters

<i>x</i>	The linklist instance.
<i>o</i>	The pointer to the item to delete.

#### See also

[linklist\\_deleteindex](#)  
[linklist\\_chuckindex](#)  
[linklist\\_chuckobject](#)

### 38.11.2.8 linklist\_findall()

```
void linklist_findall (
    t_linklist * x,
    t_linklist ** out,
    long cmpfnvoid *, void *,
    void * cmpdata )
```

Search the linked list for all items meeting defined criteria.

The items in the list are traversed, calling a specified comparison function on each, and returning the matches in another linklist.

#### Parameters

<i>x</i>	The linklist instance.
<i>out</i>	The address to a <a href="#">t_linklist</a> pointer. You should initialize the pointer to NULL before calling <a href="#">linklist_findall()</a> . A new linklist will be created internally by <a href="#">linklist_findall()</a> and returned here.
<i>cmpfn</i>	The function used to determine a match in the list.
<i>cmpdata</i>	An argument to be passed to the <a href="#">t_cmpfn</a> . This will be passed as the second of the two args to the <a href="#">t_cmpfn</a> . The first arg will be the linklist item at each iteration in the list.

## Remarks

The following example assumes you have a linklist called myLinkList, and [t\\_cmpfn](#) called myCmpFunction, and some sort of data to match in someCriteria.

```
t_linklist *results = NULL;
linklist_findall(myLinkList, &results, myCmpFunction, (void *)someCriteria);
// do something here with the 'results' linklist
// then free the results linklist
linklist_chuck(results);
```

## See also

[linklist\\_match](#)  
[t\\_cmpfn](#)  
[linklist\\_findfirst](#)

Referenced by `jit_linklist_findall()`, and `jit_linklist_findcount()`.

### 38.11.2.9 [linklist\\_findfirst\(\)](#)

```
t_atom_long linklist_findfirst (
    t_linklist * x,
    void ** o,
    long cmpfnvoid *, void *,
    void * cmpdata )
```

Search the linked list for the first item meeting defined criteria.

The items in the list are traversed, calling a specified comparison function on each until the comparison function returns true.

#### Parameters

<i>x</i>	The linklist instance.
<i>o</i>	The address to pointer that will be set with the matching item.
<i>cmpfn</i>	The function used to determine a match in the list.
<i>cmpdata</i>	An argument to be passed to the <a href="#">t_cmpfn</a> . This will be passed as the second of the two args to the <a href="#">t_cmpfn</a> . The first arg will be the linklist item at each iteration in the list.

#### Returns

The index of the matching item, or -1 if no match is found.

## Remarks

The following shows how to manually do what [linklist\\_chuckobject\(\)](#) does.

```
void *obj;
long index;
index = linklist_findfirst(x, &obj, #linklist_match, o);
if(index != -1)
    linklist_chuckindex(x, index);
```

## See also

[linklist\\_match](#)  
[t\\_cmpfn](#)  
[linklist\\_findall](#)

Referenced by `jit_linklist_findfirst()`.

**38.11.2.10 linklist\_flags()**

```
void linklist_flags (
    t_linklist * x,
    long flags )
```

Set the linklist's datastore flags.

The available flags are enumerated in [e\\_max\\_datastore\\_flags](#). These flags control the behavior of the linklist, particularly when removing items from the list using functions such as [linklist\\_clear\(\)](#), [linklist\\_deleteindex\(\)](#), or when freeing the linklist itself.

## Parameters

<i>x</i>	The linklist instance.
<i>flags</i>	A valid value from the <a href="#">e_max_datastore_flags</a> . The default is <a href="#">OBJ_FLAG_OBJ</a> .

**38.11.2.11 linklist\_funall()**

```
void linklist_funall (
    t_linklist * x,
    method fun,
    void * arg )
```

Call the specified function for every item in the linklist.

## Parameters

<i>x</i>	The linklist instance.
<i>fun</i>	The function to call, specified as function pointer cast to a Max #method.
<i>arg</i>	An argument that you would like to pass to the function being called.

## Remarks

The `linklist_funall()` method will call your function for every item in the list. It will pass both a pointer to the item in the list, and any argument that you provide. The following example shows a function that could be called by `linklist_funall()`.

```
void myFun(t_object *myObj, void *myArg)
{
    // do something with myObj and myArg here
    // myObj is the item in the linklist
}
```

### 38.11.2.12 linklist\_funall\_break()

```
t_atom_long linklist_funall_break (
    t_linklist * x,
    method fun,
    void * arg )
```

Call the specified function for every item in the linklist.

The iteration through the list will halt if the function returns a non-zero value.

## Parameters

<i>x</i>	The linklist instance.
<i>fun</i>	The function to call, specified as function pointer cast to a Max #method.
<i>arg</i>	An argument that you would like to pass to the function being called.

## Remarks

The `linklist_funall()` method will call your function for every item in the list. It will pass both a pointer to the item in the list, and any argument that you provide. The following example shows a function that could be called by `linklist_funall()`.

```
long myFun(t_symbol *myListSymbol, void *myArg)
{
    // this function is called by a linklist that contains symbols for its items
    if(myListSymbol == gensym(""))
        error("empty symbol -- aborting linklist traversal")
        return 1;
    }
    else{
        // do something with the symbol
        return 0;
    }
}
```

### 38.11.2.13 linklist\_funindex()

```
void* linklist_funindex (
    t_linklist * x,
```

```
long i,
method fun,
void * arg )
```

Call the specified function for an item specified by index.

#### Parameters

<i>x</i>	The linklist instance.
<i>i</i>	The index of the item to which to send the message.
<i>fun</i>	The function to call, specified as function pointer cast to a Max #method.
<i>arg</i>	An argument that you would like to pass to the function being called.

#### Remarks

The [linklist\\_funindex\(\)](#) method will call your function for an item in the list. It will pass both a pointer to the item in the list, and any argument that you provide. The following example shows a function that could be called by [linklist\\_funindex\(\)](#).

```
void myFun(t_object *myObj, void *myArg)
{
    // do something with myObj and myArg here
    // myObj is the item in the linklist
}
```

#### 38.11.2.14 linklist\_getflags()

```
t_atom_long linklist_getflags (
    t_linklist * x )
```

Get the linklist's datastore flags.

#### Parameters

<i>x</i>	The linklist instance.
----------	------------------------

#### Returns

The current state of the linklist flags as enumerated in [e\\_max\\_datastore\\_flags](#).

#### 38.11.2.15 linklist\_getindex()

```
void* linklist_getindex (
    t_linklist * x,
    long index )
```

Return the item stored in a linklist at a specified index.

**Parameters**

<i>x</i>	The linklist instance.
<i>index</i>	The index in the linklist to fetch. Indices are zero-based.

**Returns**

The item from the linklist stored at index. If there is no item at the index, NULL is returned

Referenced by jit\_linklist\_getindex().

**38.11.2.16 linklist\_getsize()**

```
t_atom_long linklist_getsize (
    t_linklist * x )
```

Return the number of items in a linklist object.

**Parameters**

<i>x</i>	The linklist instance.
----------	------------------------

**Returns**

The number of items in the linklist object.

Referenced by jit\_linklist\_findcount(), and jit\_linklist\_getsize().

**38.11.2.17 linklist\_insert\_sorted()**

```
long linklist_insert_sorted (
    t_linklist * x,
    void * o,
    long cmpfnvoid *, void * )
```

Insert an item into the list, keeping the list sorted according to a specified comparison function.

**Parameters**

<i>x</i>	The linklist instance.
<i>o</i>	The item pointer to insert.
<i>cmpfn</i>	A comparison function by which the list should be sorted.

**Returns**

The index of the new item in the linklist, or -1 if the insert failed.

**38.11.2.18 linklist\_insertafterobjptr()**

```
t_llitem* linklist_insertafterobjptr (
    t_linklist * x,
    void * o,
    void * objptr )
```

Insert an item into the list after another specified item.

**Parameters**

<i>x</i>	The linklist instance.
<i>o</i>	The item pointer to insert.
<i>objptr</i>	The item pointer after which to insert in the list.

**Returns**

An opaque linklist element.

**38.11.2.19 linklist\_insertbeforeobjptr()**

```
t_llitem* linklist_insertbeforeobjptr (
    t_linklist * x,
    void * o,
    void * objptr )
```

Insert an item into the list before another specified item.

**Parameters**

<i>x</i>	The linklist instance.
<i>o</i>	The item pointer to insert.
<i>objptr</i>	The item pointer before which to insert in the list.

**Returns**

An opaque linklist element.

### 38.11.2.20 linklist\_insertindex()

```
t_atom_long linklist_insertindex (
    t_linklist * x,
    void * o,
    long index )
```

Insert an item into the list at the specified index.

#### Parameters

<i>x</i>	The linklist instance.
<i>o</i>	The item pointer to insert.
<i>index</i>	The index at which to insert. Index 0 is the head of the list.

#### Returns

The index of the item in the linklist, or -1 if the insert failed.

Referenced by jit\_linklist\_insertindex().

### 38.11.2.21 linklist\_last()

```
void* linklist_last (
    t_linklist * x,
    void ** item )
```

Return the last item (the tail) in the linked-list.

#### Parameters

<i>x</i>	The linklist instance.
<i>item</i>	The address of pointer in which to store the last item in the linked-list.

#### Returns

always returns NULL

### 38.11.2.22 linklist\_makearray()

```
t_atom_long linklist_makearray (
    t_linklist * x,
```

```
void ** a,
long max )
```

Retrieve linklist items as an array of pointers.

#### Parameters

<i>x</i>	The linklist instance.
<i>a</i>	The address of the first pointer in the array to fill.
<i>max</i>	The number of pointers in the array.

#### Returns

The number of items from the list actually returned in the array.

Referenced by `jit_linklist_makearray()`.

### 38.11.2.23 `linklist_match()`

```
long linklist_match (
    void * a,
    void * b )
```

A linklist comparison method that determines if two item pointers are equal.

#### Parameters

<i>a</i>	The first item to compare.
<i>b</i>	The second item to compare.

#### Returns

Returns 1 if the items are equal, otherwise 0.

#### See also

[t\\_cmpfn](#)

### 38.11.2.24 `linklist_methodall()`

```
void linklist_methodall (
    t_linklist * x,
```

```
t_symbol * s,
... )
```

Call the named message on every object in the linklist.

The [linklist\\_methodall\(\)](#) function requires that all items in the linklist are object instances with a valid [t\\_object](#) header.

#### Parameters

<i>x</i>	The linklist instance.
<i>s</i>	The name of the message to send to the objects.
...	Any arguments to be sent with the message.

#### Remarks

Internally, this function uses [object\\_method\(\)](#), meaning that no errors will be posted if the message name does not exist for the object. It also means that messages sent methods with [A\\_GIMME](#) definitions will need to be given a symbol argument prior to the argc and argv array information.

Referenced by [jit\\_linklist\\_methodall\(\)](#).

### 38.11.2.25 linklist\_methodindex()

```
void* linklist_methodindex (
    t_linklist * x,
    t_atom_long i,
    t_symbol * s,
    ... )
```

Call the named message on an object specified by index.

The item must be an object instance with a valid [t\\_object](#) header.

#### Parameters

<i>x</i>	The linklist instance.
<i>i</i>	The index of the item to which to send the message.
<i>s</i>	The name of the message to send to the objects.
...	Any arguments to be sent with the message.

#### Remarks

Internally, this function uses [object\\_method\(\)](#), meaning that no errors will be posted if the message name does not exist for the object. It also means that messages sent methods with [A\\_GIMME](#) definitions will need to be given a symbol argument prior to the argc and argv array information.

Referenced by jit\_linklist\_methodindex().

### 38.11.2.26 linklist\_moveafterobjptr()

```
t_llelem* linklist_moveafterobjptr (
    t_linklist * x,
    void * o,
    void * objptr )
```

Move an existing item in the list to a position after another specified item in the list.

#### Parameters

<i>x</i>	The linklist instance.
<i>o</i>	The item pointer to insert.
<i>objptr</i>	The item pointer after which to move <i>o</i> in the list.

#### Returns

An opaque linklist element.

### 38.11.2.27 linklist\_movebeforeobjptr()

```
t_llelem* linklist_movebeforeobjptr (
    t_linklist * x,
    void * o,
    void * objptr )
```

Move an existing item in the list to a position before another specified item in the list.

#### Parameters

<i>x</i>	The linklist instance.
<i>o</i>	The item pointer to insert.
<i>objptr</i>	The item pointer before which to move <i>o</i> in the list.

#### Returns

An opaque linklist element.

### 38.11.2.28 linklist\_new()

```
BEGIN_USING_C_LINKAGE t_linklist* linklist_new (
    void )
```

Create a new linklist object.

You can free the linklist by calling [object\\_free\(\)](#) on the linklist's pointer, or by using [linklist\\_chuck\(\)](#).

#### Returns

Pointer to the new linklist object.

#### See also

[object\\_free\(\)](#)  
[linklist\\_chuck\(\)](#)

Referenced by [jit\\_linklist\\_new\(\)](#).

### 38.11.2.29 linklist\_next()

```
void* linklist_next (
    t_linklist * x,
    void * p,
    void ** next )
```

Given an item in the list, find the next item.

This provides an means for walking the list.

#### Parameters

<i>x</i>	The linklist instance.
<i>p</i>	An item in the list.
<i>next</i>	The address of a pointer to set with the next item in the list.

### 38.11.2.30 linklist\_objptr2index()

```
t_atom_long linklist_objptr2index (
    t_linklist * x,
    void * p )
```

Return an item's index, given the item itself.

**Parameters**

<i>x</i>	The linklist instance.
<i>p</i>	The item pointer to search for in the linklist.

**Returns**

The index of the item given in the linklist. If the item is not in the linklist [MAX\\_ERR\\_GENERIC](#) is returned.

Referenced by `jit_linklist_objptr2index()`.

**38.11.2.31 linklist\_prev()**

```
void* linklist_prev (
    t_linklist * x,
    void * p,
    void ** prev )
```

Given an item in the list, find the previous item.

This provides an means for walking the list.

**Parameters**

<i>x</i>	The linklist instance.
<i>p</i>	An item in the list.
<i>prev</i>	The address of a pointer to set with the previous item in the list.

**38.11.2.32 linklist\_READONLY()**

```
void linklist_READONLY (
    t_linklist * x,
    long readonly )
```

Set the linklist's readonly bit.

By default the readonly bit is 0, indicating that it is threadsafe for both reading and writing. Setting the readonly bit to 1 will disable the linklist's threadsafety mechanism, increasing performance but at the expense of threadsafe operation. Unless you can guarantee the threading context for a linklist's use, you should leave this set to 0.

**Parameters**

<i>x</i>	The linklist instance.
<i>readonly</i>	A 1 or 0 for setting the readonly bit.

### 38.11.2.33 linklist\_reverse()

```
void linklist_reverse (
    t_linklist * x )
```

Reverse the order of items in the linked-list.

#### Parameters

x	The linklist instance.
---	------------------------

Referenced by jit\_linklist\_reverse().

### 38.11.2.34 linklist\_rotate()

```
void linklist_rotate (
    t_linklist * x,
    long i )
```

Rotate items in the linked list in circular fashion.

#### Parameters

x	The linklist instance.
i	The number of positions in the list to shift items.

Referenced by jit\_linklist\_rotate().

### 38.11.2.35 linklist\_shuffle()

```
void linklist_shuffle (
    t_linklist * x )
```

Randomize the order of items in the linked-list.

#### Parameters

x	The linklist instance.
---	------------------------

Referenced by jit\_linklist\_shuffle().

### 38.11.2.36 linklist\_sort()

```
void linklist_sort (
    t_linklist * x,
    long cmpfnvoid *, void * )
```

Sort the linked list.

The items in the list are ordered using a [t\\_cmpfn](#) function that is passed in as an argument.

#### Parameters

x	The linklist instance.
cmpfn	The function used to sort the list.

#### Remarks

The following example is a real-world example of sorting a linklist of symbols alphabetically by first letter only.

First the cmpfn is defined, then it is used in a different function by [linklist\\_sort\(\)](#).

```
long myAlphabeticalCmpfn(void *a, void *b)
{
    t_symbol *s1 = (t_symbol *)a;
    t_symbol *s2 = (t_symbol *)b;
    if(s1->s_name[0] < s2->s_name[0])
        return true;
    else
        return false;
}
void mySortMethod(t_myobj *x)
{
    // the linklist was already created and filled with items previously
    linklist_sort(x->myLinkList, myAlphabeticalCmpfn);
}
```

Referenced by jit\_linklist\_sort().

### 38.11.2.37 linklist\_substitute()

```
void* linklist_substitute (
    t_linklist * x,
    void * p,
    void * newp )
```

Given an item in the list, replace it with a different value.

**Parameters**

<i>x</i>	The linklist instance.
<i>p</i>	An item in the list.
<i>newp</i>	The new value.

**Returns**

Always returns NULL.

**38.11.2.38 linklist\_swap()**

```
void linklist_swap (
    t_linklist * x,
    long a,
    long b )
```

Swap the position of two items in the linked-list, specified by index.

**Parameters**

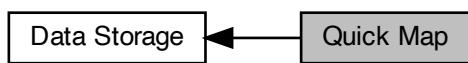
<i>x</i>	The linklist instance.
<i>a</i>	The index of the first item to swap.
<i>b</i>	The index of the second item to swap.

Referenced by jit\_linklist\_swap().

**38.12 Quick Map**

A quickmap implements a pair of [t\\_hashtab](#) hash tables so that it is fast to look up a unique value for a unique key or vice-versa.

Collaboration diagram for Quick Map:



## Data Structures

- struct `t_quickmap`

*The quickmap object.*

## Functions

- `BEGIN_USING_C_LINKAGE void * quickmap_new (void)`  
*Create a new quickmap object.*
- `void quickmap_add (t_quickmap *x, void *p1, void *p2)`  
*Add a pair of keys mapped to each other to the quickmap.*
- `void quickmap_drop (t_quickmap *x, void *p1, void *p2)`  
*Drop a pair of keys mapped to each other in the quickmap.*
- `long quickmap_lookup_key1 (t_quickmap *x, void *p1, void **p2)`  
*Given a (first) key, lookup the value (the second key).*
- `long quickmap_lookup_key2 (t_quickmap *x, void *p1, void **p2)`  
*Given a (second) key, lookup the value (the first key).*
- `void quickmap_READONLY (t_quickmap *x, long way)`  
*Set the readonly flag of the quickmap's hash tables.*

### 38.12.1 Detailed Description

A quickmap implements a pair of `t_hashtab` hash tables so that it is fast to look up a unique value for a unique key or vice-versa.

This implies that both the keys and the values must be unique so that look-ups can be performed in both directions.

### 38.12.2 Function Documentation

#### 38.12.2.1 `quickmap_add()`

```
void quickmap_add (
    t_quickmap * x,
    void * p1,
    void * p2 )
```

Add a pair of keys mapped to each other to the quickmap.

Note that these are considered to be a `t_symbol` internally. This means that if you are mapping a `t_symbol` to a `t_object`, for example, the `t_object` will not automatically be freed when you free the quickmap (unlike what happens when you typically free a `t_hashtab`).

**Parameters**

<i>x</i>	The quickmap instance.
<i>p1</i>	The (first) key.
<i>p2</i>	The value (or the second key).

**Returns**

A Max error code.

**38.12.2.2 quickmap\_drop()**

```
void quickmap_drop (
    t_quickmap * x,
    void * p1,
    void * p2 )
```

Drop a pair of keys mapped to each other in the quickmap.

**Parameters**

<i>x</i>	The quickmap instance.
<i>p1</i>	The first key.
<i>p2</i>	The second key.

**Returns**

A Max error code.

**38.12.2.3 quickmap\_lookup\_key1()**

```
long quickmap_lookup_key1 (
    t_quickmap * x,
    void * p1,
    void ** p2 )
```

Given a (first) key, lookup the value (the second key).

**Parameters**

<i>x</i>	The quickmap instance.
<i>p1</i>	The (first) key.
<i>p2</i>	The address of a pointer which will hold the resulting key upon return.

**Returns**

A Max error code.

**38.12.2.4 quickmap\_lookup\_key2()**

```
long quickmap_lookup_key2 (
    t_quickmap * x,
    void * p1,
    void ** p2 )
```

Given a (second) key, lookup the value (the first key).

**Parameters**

<i>x</i>	The quickmap instance.
<i>p1</i>	The (second) key.
<i>p2</i>	The address of a pointer which will hold the resulting key upon return.

**Returns**

A Max error code.

**38.12.2.5 quickmap\_new()**

```
BEGIN_USING_C_LINKAGE void* quickmap_new (
    void )
```

Create a new quickmap object.

**Returns**

Pointer to the new quickmap object.

**38.12.2.6 quickmap\_READONLY()**

```
void quickmap_READONLY (
    t_quickmap * x,
    long way )
```

Set the readonly flag of the quickmap's hash tables.

See [hashtab\\_READONLY\(\)](#) for more information about this.

**Parameters**

<code>x</code>	The quickmap instance.
<code>way</code>	Set to true to make the quickmap readonly (disable thread protection) or false (the default) to enable thread protection.

## 38.13 String Object

Max's string object is a simple wrapper for c-strings, useful when working with Max's `t_dictionary`, `t_linklist`, or `t_hashtab`.

Collaboration diagram for String Object:



### Data Structures

- struct `t_string`  
*The string object.*

### Functions

- `t_string * string_new (const char *psz)`  
*Create a new string object.*
- `const char * string_getptr (t_string *x)`  
*Fetch a pointer to a string object's internal C-string.*
- `void string_reserve (t_string *x, long numbytes)`  
*Reserve additional memory for future string growth.*
- `void string_append (t_string *x, const char *s)`  
*Append a C-string onto the existing string maintained by a `t_string` object.*
- `void string_chop (t_string *x, long numchars)`  
*Shorten a string by eliminating N characters from the end.*

#### 38.13.1 Detailed Description

Max's string object is a simple wrapper for c-strings, useful when working with Max's `t_dictionary`, `t_linklist`, or `t_hashtab`.

#### See also

[Dictionary](#)

### 38.13.2 Function Documentation

#### 38.13.2.1 string\_append()

```
void string_append (
    t_string * x,
    const char * s )
```

Append a C-string onto the existing string maintained by a `t_string` object.

Memory allocation for the string will grow as needed to hold the concatenated string.

##### Parameters

<code>x</code>	The string object instance.
<code>s</code>	A string to append/concatenate with the existing string.

#### 38.13.2.2 string\_chop()

```
void string_chop (
    t_string * x,
    long numchars )
```

Shorten a string by eliminating N characters from the end.

##### Parameters

<code>x</code>	The string object instance.
<code>numchars</code>	The number of characters to chop from the end of the string.

#### 38.13.2.3 string\_getptr()

```
const char* string_getptr (
    t_string * x )
```

Fetch a pointer to a string object's internal C-string.

**Parameters**

x	The string object instance.
---	-----------------------------

**Returns**

A pointer to the internally maintained C-string.

**38.13.2.4 string\_new()**

```
t_string* string_new (
    const char * psz )
```

Create a new string object.

**Parameters**

psz	Pointer to a C-string that will be copied to memory internal to this string object instance.
-----	--

**Returns**

The new string object instance pointer.

**38.13.2.5 string\_reserve()**

```
void string_reserve (
    t_string * x,
    long numbytes )
```

Reserve additional memory for future string growth.

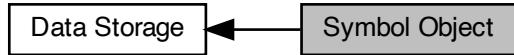
**Parameters**

x	The string object instance.
numbytes	The total number of bytes to allocate for this string object.

**38.14 Symbol Object**

The symobject class is a simple object that wraps a `t_symbol*` together with a couple of additional fields.

Collaboration diagram for Symbol Object:



## Data Structures

- struct [t\\_symobject](#)  
*The symobject data structure.*

## Functions

- [BEGIN\\_USING\\_C\\_LINKAGE void \\* symobject\\_new \(t\\_symbol \\*sym\)](#)  
*The symobject data structure.*
- long [symobject\\_linklist\\_match \(void \\*a, void \\*b\)](#)  
*Utility for searching a linklist containing symobjects.*

### 38.14.1 Detailed Description

The symobject class is a simple object that wraps a [t\\_symbol\\*](#) together with a couple of additional fields.

It is useful for storing symbols, possibly with additional flags or pointers, into a [Hash Table](#) or [Linked List](#).

#### Version

5.0

### 38.14.2 Function Documentation

#### 38.14.2.1 [symobject\\_linklist\\_match\(\)](#)

```
long symobject_linklist_match (
    void * a,
    void * b )
```

Utility for searching a linklist containing symobjects.

**Parameters**

<i>a</i>	(opaque)
<i>b</i>	(opaque)

**Returns**

Returns true if a match is found, otherwise returns false.

**Remarks**

The following example shows one common use of the this method.

```
t_symobject *item = NULL;
long index;
t_symbol *textsym;
texts sym = gensym("something to look for");
// search for a symobject with the symbol 'something to look for'
index = linklist_findfirst(s_ll_history, (void **)&item, symobject_linklist_match, textsym);
if(index == -1){
    // symobject not found.
}
else{
    do something with the symobject, or with the index of the symobject in the linklist
}
```

**38.14.2.2 symobject\_new()**

```
BEGIN_USING_C_LINKAGE void* symobject_new (
    t_symbol * sym )
```

The symobject data structure.

**Parameters**

<i>sym</i>	A symbol with which to initialize the new symobject.
------------	--

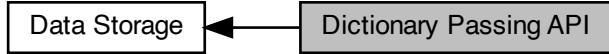
**Returns**

Pointer to the new symobject instance.

**38.15 Dictionary Passing API**

The Dictionary Passing API defines a means by which [t\\_dictionary](#) instances may be passed between Max objects in a way similar to the way Jitter Matrices are passed between objects.

Collaboration diagram for Dictionary Passing API:



## Functions

- **BEGIN\_USING\_C\_LINKAGE t\_dictionary \* dictobj\_register (t\_dictionary \*d, t\_symbol \*\*name)**  
*Register a `t_dictionary` with the dictionary passing system and map it to a unique name.*
- **t\_max\_err dictobj\_unregister (t\_dictionary \*d)**  
*Unregister a `t_dictionary` with the dictionary passing system.*
- **t\_dictionary \* dictobj\_findregistered\_clone (t\_symbol \*name)**  
*Find the `t_dictionary` for a given name, and return a copy of that dictionary. When you are done, do not call `dictobj_release()` on the dictionary, because you are working on a copy rather than on a retained pointer.*
- **t\_dictionary \* dictobj\_findregistered\_retain (t\_symbol \*name)**  
*Find the `t_dictionary` for a given name, return a pointer to that `t_dictionary`, and increment its reference count.*
- **t\_max\_err dictobj\_release (t\_dictionary \*d)**  
*For a `t_dictionary`/name that was previously retained with `dictobj_findregistered_retain()`, release it (decrement its reference count).*
- **t\_symbol \* dictobj\_namefromptr (t\_dictionary \*d)**  
*Find the name associated with a given `t_dictionary`.*
- **void dictobj\_outlet\_atoms (void \*out, long argc, t\_atom \*argv)**  
*Send atoms to an outlet in your Max object, handling complex datatypes that may be present in those atoms.*
- **long dictobj\_atom\_safety (t\_atom \*a)**  
*Ensure that an atom is safe for passing.*
- **long dictobj\_atom\_safety\_flags (t\_atom \*a, long flags)**  
*Ensure that an atom is safe for passing.*
- **long dictobj\_validate (const t\_dictionary \*schema, const t\_dictionary \*candidate)**  
*Validate the contents of a `t_dictionary` against a second `t_dictionary` containing a schema.*
- **t\_max\_err dictobj\_jsonfromstring (long \*jsonsize, char \*\*json, const char \*str)**  
*Convert a C-string of `Dictionary Syntax` into a C-string of JSON.*
- **t\_max\_err dictobj\_dictionaryfromstring (t\_dictionary \*\*d, const char \*str, int str\_is\_already\_json, char \*errorstring)**  
*Create a new `t_dictionary` from `Dictionary Syntax` which is passed in as a C-string.*
- **t\_max\_err dictobj\_dictionaryfromatoms (t\_dictionary \*\*d, const long argc, const t\_atom \*argv)**  
*Create a new `t_dictionary` from `Dictionary Syntax` which is passed in as an array of atoms.*
- **t\_max\_err dictobj\_dictionaryfromatoms\_extended (t\_dictionary \*\*d, const t\_symbol \*msg, long argc, const t\_atom \*argv)**  
*Create a new `t_dictionary` from an array of atoms that use Max dictionary syntax, JSON, or compressed JSON.*
- **t\_max\_err dictobj\_dictionarytoatoms (const t\_dictionary \*d, long \*argc, t\_atom \*\*argv)**  
*Serialize the contents of a `t_dictionary` into `Dictionary Syntax`.*

- `t_max_err dictobj_key_parse (t_object *x, t_dictionary *d, t_atom *akey, t_bool create, t_dictionary **targetdict, t_symbol **targetkey, t_int32 *index)`

*Given a complex key (one that includes potential heirarchy or array-member access), return the actual key and the dictionary in which the key should be referenced.*

### 38.15.1 Detailed Description

The Dictionary Passing API defines a means by which `t_dictionary` instances may be passed between Max objects in a way similar to the way Jitter Matrices are passed between objects.

There are important differences, however, between Jitter matrix passing and dictionary passing. Many of these differences are documented in Max's documentation on dictionaries and structured data.

Every dictionary instance in this system is mapped to a unique name that identifies the dictionary. Dictionaries are passed between objects using the "dictionary" message with a single argument, which is the name of the dictionary.

### 38.15.2 Registration and Access

The C-API for working with these dictionaries is composed of 5 primary registration/access methods:

- `dictobj_register()` : register a `t_dictionary` instance with the system, and map the instance to a name
- `dictobj_unregister()` : unregister a `t_dictionary` from the system
- `dictobj_findregistered_clone()` : find the `t_dictionary` for a given name, and return a *copy* of that dictionary
- `dictobj_findregistered_retain()` : find the `t_dictionary` for a given name, return a pointer to that `t_dictionary`, and increment its reference count.
- `dictobj_release()` : for a `t_dictionary`/name that was previously retained with `dictobj_findregistered_retain()`, release it (decrement its reference count).

It is useful to think of objects in the dictionary system as "nouns" and "verbs".

A "noun" is an object that possess or owns a dictionary. These objects are servers whose dictionary will accessed by other object that are clients. An example of a "noun" is the dict.pack object that creates a dictionary that is passed to other objects.

A "verb" is an object that does not maintain its own dictionary (it is not a thing) but merely does something to any dictionaries it receives. This object is a client rather than a server. An example of a "verb" is the dict.strip object, which removes entries from an existing dictionary but possesses no dictionary of its own.

Any object which is a dictionary "noun", can keep and rely on their dictionary pointer. Because of the way `object_register()` works, there should be no possibility for this pointer to change behind the scenes. They each need to call `object_free()` on their respective object pointer, however. A call to `object_free()` also calls `object_unregister()` once, so there's technically not a need to unregister from the owner itself. They work like jit.matrix (and similar to buffer~), and use `object_register()` to increment a server reference count. If an object has already registered an object with the given name, the pointer passed in to register is freed and the existing one is returned from the registration function.

Dictionary "verbs" on the other hand should just call `dict_findregistered_retain()` and `dict_release()` when done. They are not incrementing the server reference count. They increment a reference count with regards to object freeing, which is compatible with and complementary to the server reference count.

### 38.15.3 Dictionary Syntax

Dictionaries may be represented in a variety of textual formats including JSON. Max also supports a compact YAML-like dictionary notation which is useful for proving data structure contents as lists of atoms in object boxes. This format is documented in Max's documentation of the dictionary features. The following functions are used for formatting and parsing the dictionary syntax.

- [dictobj\\_jsonfromstring\(\)](#)
- [dictobj\\_dictionaryfromstring\(\)](#)
- [dictobj\\_dictionaryfromatoms\(\)](#)
- [dictobj\\_dictionarytoatoms\(\)](#)

### 38.15.4 Utilities

There are several utility functions available to assist in coding objects that pass dictionaries.

- [dictobj\\_outlet\\_atoms\(\)](#)
- [dictobj\\_atom\\_safety\(\)](#)
- [dictobj\\_validate\(\)](#)

The [dictobj\\_validate\(\)](#) object is a utility routine for validating a dictionary against "schema" dictionary. This enables a behavior somewhat analogous to Objective-C or Smalltalk prototypes. Dictionary validation can be useful to implement a kind of dictionary polymorphism. For a multiple-inheritance behavior, simply validate a dictionary against multiple schemas to verify the presence of required keys and values.

### 38.15.5 Limitations

The [dict\\_outlet\\_atoms\(\)](#) function will not output [A\\_OBJ](#) atoms directly (nor should any other object) and as such it will also not output [t\\_atomarray](#) instances containing objects, thus atomarrays are *not* hierarchical in the dictionary passing implementation.

It will output an atom array if provided a single [A\\_OBJ](#) atom with class atomarray. If there is an array of atoms which contain [A\\_OBJ](#) atoms, they are converted to the \*symbols\* <dictionary-object>, <atomarray-object>, <string-object>, <other-object> respectively. Ideally such a case should never be reached if everything which inserts values into a dictionary is well behaved-i.e.

- a key may be a single atom
- a key may be an atomarray (but no [A\\_OBJ](#) atoms)
- a key may be a dictionary

---

#### Version

6.0

### 38.15.6 Function Documentation

#### 38.15.6.1 dictobj\_atom\_safety()

```
long dictobj_atom_safety (
    t_atom * a )
```

Ensure that an atom is safe for passing.

Atoms are allowed to be [A\\_LONG](#), [A\\_FLOAT](#), or [A\\_SYM](#), but not [A\\_OBJ](#). If the atom is an [A\\_OBJ](#), it will be converted into something that will be safe to pass.

##### Parameters

<i>a</i>	An atom to check, and potentially modify, to ensure safety in the dictionary-passing system.
----------	--

##### Returns

If the atom was changed then 1 is returned. Otherwise 0 is returned.

#### 38.15.6.2 dictobj\_atom\_safety\_flags()

```
long dictobj_atom_safety_flags (
    t_atom * a,
    long flags )
```

Ensure that an atom is safe for passing.

Atoms are allowed to be [A\\_LONG](#), [A\\_FLOAT](#), or [A\\_SYM](#), but not [A\\_OBJ](#). If the atom is an [A\\_OBJ](#), it will be converted into something that will be safe to pass.

##### Parameters

<i>a</i>	An atom to check, and potentially modify, to ensure safety in the dictionary-passing system.
<i>flags</i>	Pass DICTOBJ_ATOM_FLAGS_REGISTER to have dictionary atoms registered/retained.

##### Returns

If the atom was changed then 1 is returned. Otherwise 0 is returned.

### 38.15.6.3 dictobj\_dictionaryfromatoms()

```
t_max_err dictobj_dictionaryfromatoms (
    t_dictionary ** d,
    const long argc,
    const t_atom * argv )
```

Create a new [t\\_dictionary](#) from [Dictionary Syntax](#) which is passed in as an array of atoms.

Unlike many [t\\_dictionary](#) calls to create dictionaries, this function does not take over ownership of the atoms you pass in.

#### Parameters

<i>d</i>	The address of a dictionary variable, which will hold a pointer to the new dictionary upon return. Should be initialized to NULL.
<i>argc</i>	The number of atoms in argv.
<i>argv</i>	Pointer to the first of an array of atoms to be interpreted as <a href="#">Dictionary Syntax</a> .

#### Returns

A Max error code.

#### See also

[dictobj\\_dictionaryfromatoms\\_extended\(\)](#) [dictobj\\_dictionarytoatoms\(\)](#)

### 38.15.6.4 dictobj\_dictionaryfromatoms\_extended()

```
t_max_err dictobj_dictionaryfromatoms_extended (
    t_dictionary ** d,
    const t_symbol * msg,
    long argc,
    const t_atom * argv )
```

Create a new [t\\_dictionary](#) from from an array of atoms that use Max dictionary syntax, JSON, or compressed JSON.

This function is the C analog to the `dict.deserialize` object in Max. Unlike many [t\\_dictionary](#) calls to create dictionaries, this function does not take over ownership of the atoms you pass in.

#### Parameters

<i>d</i>	The address of a dictionary variable, which will hold a pointer to the new dictionary upon return. Should be initialized to NULL.
<i>msg</i>	Ignored.
<i>argc</i>	The number of atoms in argv.
<i>argv</i>	Pointer to the first of an array of atoms to be interpreted as <a href="#">Dictionary Syntax</a> , JSON, or compressed JSON.

**Returns**

A Max error code.

**See also**

[dictobj\\_dictionaryfromatoms\(\)](#) [dictobj\\_dictionaryfromstring\(\)](#)

**38.15.6.5 dictobj\_dictionaryfromstring()**

```
t_max_err dictobj_dictionaryfromstring (
    t_dictionary ** d,
    const char * str,
    int str_is_already_json,
    char * errorstring )
```

Create a new `t_dictionary` from [Dictionary Syntax](#) which is passed in as a C-string.

**Parameters**

<code>d</code>	The address of a dictionary variable, which will hold a pointer to the new dictionary upon return. Should be initialized to NULL.
<code>str</code>	A NULL-terminated C-string containing <a href="#">Dictionary Syntax</a> .
<code>str_is_already_json</code>	
<code>errorstring</code>	

**Returns**

A Max error code.

**See also**

[dictobj\\_dictionarytoatoms\(\)](#)

**38.15.6.6 dictobj\_dictionarytoatoms()**

```
t_max_err dictobj_dictionarytoatoms (
    const t_dictionary * d,
    long * argc,
    t_atom ** argv )
```

Serialize the contents of a `t_dictionary` into [Dictionary Syntax](#) .

**Parameters**

<i>d</i>	The dictionary to serialize.
<i>argc</i>	The address of a variable to hold the number of atoms allocated upon return.
<i>argv</i>	The address of a <a href="#">t_atom</a> pointer which will point to the first atom (of an array of argc atoms) upon return. You are responsible for freeing the pointer returned with <a href="#">sysmem_freeptr()</a> when you are done with it. If you pass in existing memory, it must be memory allocated using <a href="#">sysmem_newptr()</a> and the pointer may be resized.

**Returns**

A Max error code.

**See also**

[dictobj\\_dictionaryfromatoms\(\)](#)

**38.15.6.7 dictobj\_findregistered\_clone()**

```
t_dictionary* dictobj_findregistered_clone (
    t_symbol * name )
```

Find the [t\\_dictionary](#) for a given name, and return a *copy* of that dictionary. When you are done, do *not* call [dictobj\\_release\(\)](#) on the dictionary, because you are working on a copy rather than on a retained pointer.

**Parameters**

<i>name</i>	The name associated with the dictionary for which you wish to obtain a copy.
-------------	--

**Returns**

The dictionary cloned from the existing dictionary. Returns NULL if no dictionary is associated with name.

**See also**

[dictobj\\_findregistered\\_retain\(\)](#)

**38.15.6.8 dictobj\_findregistered\_retain()**

```
t_dictionary* dictobj_findregistered_retain (
    t_symbol * name )
```

Find the [t\\_dictionary](#) for a given name, return a pointer to that [t\\_dictionary](#), and increment its reference count.

When you are done you should call [dictobj\\_release\(\)](#) on the dictionary.

**Parameters**

<i>name</i>	The name associated with the dictionary for which you wish to obtain a pointer.
-------------	---

**Returns**

A pointer to the dictionary associated with name. Returns NULL if no dictionary is associated with name.

**See also**

[dictobj\\_release\(\)](#)  
[dictobj\\_findregistered\\_clone\(\)](#)

**38.15.6.9 dictobj\_jsonfromstring()**

```
t_max_err dictobj_jsonfromstring (
    long * jsonsize,
    char ** json,
    const char * str )
```

Convert a C-string of [Dictionary Syntax](#) into a C-string of JSON.

**Parameters**

<i>jsonsize</i>	The address of a variable to be filled-in with the number of chars in json upon return.
<i>json</i>	The address of a char pointer to point to the JSON C-string upon return. Should be initialized to NULL. You are responsible for freeing the string with <a href="#">sysmem_freeptr()</a> when you are done with it.
<i>str</i>	A NULL-terminated C-string containing <a href="#">Dictionary Syntax</a> .

**Returns**

A Max error code.

**See also**

[dictobj\\_dictionarytoatoms\(\)](#)

**38.15.6.10 dictobj\_key\_parse()**

```
t_max_err dictobj_key_parse (
    t_object * x,
```

```
t_dictionary * d,
t_atom * akey,
t_bool create,
t_dictionary ** targetdict,
t_symbol ** targetkey,
t_int32 * index )
```

Given a complex key (one that includes potential heirarchy or array-member access), return the actual key and the dictionary in which the key should be referenced.

#### Parameters

<i>x</i>	Your calling object. If there is an error this will be used by the internal call to <a href="#">object_error()</a> .
<i>d</i>	The dictionary you are querying.
<i>akey</i>	The complex key specifying the query.
<i>create</i>	If true, create the intermediate dictionaries in the hierarchy specified in <i>akey</i> .
<i>targetdict</i>	Returns the <a href="#">t_dictionary</a> that for the (sub)dictionary specified by <i>akey</i> .
<i>targetkey</i>	Returns the name of the key in <i>targetdict</i> that to which <i>akey</i> is referencing.
<i>index</i>	Returns the index requested if array-member access is specified. Pass NULL if you are not interested in this.

#### Returns

A Max error code.

### 38.15.6.11 dictobj\_namefromptr()

```
t_symbol* dictobj_namefromptr (
    t_dictionary * d )
```

Find the name associated with a given [t\\_dictionary](#).

#### Parameters

<i>d</i>	A dictionary, whose name you wish to determine.
----------	---

#### Returns

The symbol associated with the dictionary, or NULL if the dictionary is not registered.

#### See also

[dictobj\\_register\(\)](#)

### 38.15.6.12 dictobj\_outlet\_atoms()

```
void dictobj_outlet_atoms (
    void * out,
    long argc,
    t_atom * argv )
```

Send atoms to an outlet in your Max object, handling complex datatypes that may be present in those atoms.

This is particularly when sending the contents of a dictionary entry out of an outlet as in the following example code.

```
long ac = 0;
t_atom *av = NULL;
t_max_err err;
err = dictionary_copyatoms(d, key, &ac, &av);
if (!err && ac && av) {
    // handles singles, lists, symbols, atomarrays, dictionaries, etc.
    dictobj_outlet_atoms(x->outlets[i], ac, av);
}
if (av)
    sysmem_freeptr(av);
```

#### Parameters

<i>out</i>	The outlet through which the atoms should be sent.
<i>argc</i>	The count of atoms in <i>argv</i> .
<i>argv</i>	Pointer to the first of an array of atoms to send to the outlet.

### 38.15.6.13 dictobj\_register()

```
BEGIN_USING_C_LINKAGE t_dictionary* dictobj_register (
    t_dictionary * d,
    t_symbol ** name )
```

Register a [t\\_dictionary](#) with the dictionary passing system and map it to a unique name.

#### Parameters

<i>d</i>	A valid dictionary object.
<i>name</i>	The address of a <a href="#">t_symbol</a> pointer to the name you would like mapped to this dictionary. If the <a href="#">t_symbol</a> pointer has a NULL value then a unique name will be generated and filled-in upon return.

#### Returns

The dictionary mapped to the specified name.

### 38.15.6.14 dictobj\_release()

```
t_max_err dictobj_release (
    t_dictionary * d )
```

For a [t\\_dictionary](#)/name that was previously retained with [dictobj\\_findregistered\\_retain\(\)](#), release it (decrement its reference count).

#### Parameters

<code>d</code>	A valid dictionary object retained by <a href="#">dictobj_findregistered_retain()</a> .
----------------	---

#### Returns

A Max error code.

#### See also

[dictobj\\_findregistered\\_retain\(\)](#)

### 38.15.6.15 dictobj\_unregister()

```
t_max_err dictobj_unregister (
    t_dictionary * d )
```

Unregister a [t\\_dictionary](#) with the dictionary passing system.

Generally speaking you should not need to call this method. Calling [object\\_free\(\)](#) on the [t\\_dictionary](#) automatically unregisters it.

#### Parameters

<code>d</code>	A valid dictionary object.
----------------	----------------------------

#### Returns

A Max error code.

### 38.15.6.16 dictobj\_validate()

```
long dictobj_validate (
    const t_dictionary * schema,
    const t_dictionary * candidate )
```

Validate the contents of a [t\\_dictionary](#) against a second [t\\_dictionary](#) containing a schema.

The schema dictionary contains keys and values, like any dictionary. [dictobj\\_validate\(\)](#) checks to make sure that all keys in the schema dictionary are present in the candidate dictionary. If the keys are all present then the candidate passes and the function returns true. Otherwise the the candidate fails the validation and the function returns false.

Generally speaking, the schema dictionary will contain values with the symbol "\*", indicating a wildcard, and thus only the key is used to validate the dictionary (all values match the wildcard). However, if the schema dictionary contains non-wildcard values for any of its keys, those keys in the candidate dictionary must also contain matching values in order for the candidate to successfully validate.

An example of this in action is the dict.route object in Max, which simply wraps this function.

#### Parameters

<i>schema</i>	The dictionary against which to validate candidate.
<i>candidate</i>	A dictionary to test against the schema.

#### Returns

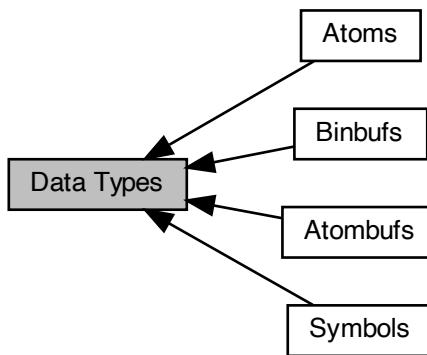
Returns true if the candidate validates against the schema, otherwise returns false.

#### See also

[dictobj\\_dictionarytoatoms\(\)](#)

## 38.16 Data Types

Collaboration diagram for Data Types:



## Modules

- [Atoms](#)
- [Atombufs](#)

*An Atombuf is an alternative to [Binbufs](#) for temporary storage of atoms.*

- [Binbufs](#)

*You won't need to know about the internal structure of a Binbuf, so you can use the void \* type to refer to one.*

- [Symbols](#)

*Max maintains a symbol table of all strings to speed lookup for message passing.*

## Data Structures

- struct [t\\_rect](#)

*Coordinates for specifying a rectangular region.*

- struct [t\\_pt](#)

*Coordinates for specifying a point.*

- struct [t\\_size](#)

*Coordinates for specifying the size of a region.*

## Macros

- #define [C74\\_TYPEOF](#)

*Function pointer type for generic methods.*

## Typedefs

- typedef long(\* [t\\_intmethod](#)) (void \*)

*Function pointer type for methods returning a long.*

- typedef void \*(\*[zero\\_meth](#)) (void \*x)

*Function pointer type for methods with no arguments.*

- typedef void \*(\*[one\\_meth](#)) (void \*x, void \*z)

*Function pointer type for methods with a single argument.*

- typedef void \*(\*[two\\_meth](#)) (void \*x, void \*z, void \*a)

*Function pointer type for methods with two arguments.*

- typedef long \*(\*[gimmeback\\_meth](#)) (void \*x, [t\\_symbol](#) \*s, long ac, [t\\_atom](#) \*av, [t\\_atom](#) \*rv)

*Function pointer type for methods that pass back a result value through the last parameter as a [t\\_atom](#), and return an error.*

### 38.16.1 Detailed Description

## 38.17 Atoms

Collaboration diagram for Atoms:



### Data Structures

- union `word`  
*Union for packing any of the datum defined in `e_max_atomtypes`.*
- struct `t_atom`  
*An atom is a typed datum.*

### Enumerations

- enum `e_max_atomtypes` {
 `A_NOTHING, A_LONG, A_FLOAT, A_SYM,`  
`A_OBJ, A_DEFLONG, A_DEFFLOAT, A_DEFSYM,`  
`A_GIMME, A_CANT, A_SEMI, A_COMMA,`  
`A_DOLLAR, A_DOLLSYM, A_GIMMEBACK, A_DEFER,`  
`A_USURP, A_DEFER_LOW, A_USURP_LOW }`

*the list of officially recognized types, including pseudotypes for commas and semicolons.*

- enum  
*Defines the largest possible string size for an atom.*
- enum `e_max_atom_gettext_flags` {
 `OBEX_UTIL_ATOM_GETTEXT_DEFAULT, OBEX_UTIL_ATOM_GETTEXT_TRUNCATE_ZEROS, OBEX_UTIL_ATOM_GETTEXT,`  
`, OBEX_UTIL_ATOM_GETTEXT_SYM_FORCE_QUOTE,`  
`OBEX_UTIL_ATOM_GETTEXT_COMMAS_DELIM, OBEX_UTIL_ATOM_GETTEXT_FORCE_ZEROS, OBEX_UTIL_ATOM_GETTEXT,`  
`, OBEX_UTIL_ATOM_GETTEXT_NUM_LO_RES,`  
`OBEX_UTIL_ATOM_GETTEXT_NOESCAPE, OBEX_UTIL_ATOM_GETTEXT_LINEBREAK_NODELIM }`

*Flags that determine how functions convert atoms into text (C-strings).*

## Functions

- `t_max_err atom_setlong (t_atom *a, t_atom_long b)`  
`Inserts an integer into a t_atom and change the t_atom's type to A_LONG.`
- `t_max_err atom_setfloat (t_atom *a, double b)`  
`Inserts a floating point number into a t_atom and change the t_atom's type to A_FLOAT.`
- `t_max_err atom_setsym (t_atom *a, t_symbol *b)`  
`Inserts a t_symbol* into a t_atom and change the t_atom's type to A_SYM.`
- `t_max_err atom_setobj (t_atom *a, void *b)`  
`Inserts a generic pointer value into a t_atom and change the t_atom's type to A_OBJ.`
- `t_atom_long atom_getlong (const t_atom *a)`  
`Retrieves a long integer value from a t_atom.`
- `t_atom_float atom_getfloat (const t_atom *a)`  
`Retrieves a floating point value from a t_atom.`
- `t_symbol * atom_getsym (const t_atom *a)`  
`Retrieves a t_symbol* value from a t_atom.`
- `void * atom_getobj (const t_atom *a)`  
`Retrieves a generic pointer value from a t_atom.`
- `long atom_getcharfix (const t_atom *a)`  
`Retrieves an unsigned integer value between 0 and 255 from a t_atom.`
- `long atom_gettype (const t_atom *a)`  
`Retrieves type from a t_atom.`
- `t_max_err atom_arg_getlong (t_atom_long *c, long idx, long ac, const t_atom *av)`  
`Retrieves the integer value of a particular t_atom from an atom list, if the atom exists.`
- `long atom_arg_getfloat (float *c, long idx, long ac, const t_atom *av)`  
`Retrieves the floating point value of a particular t_atom from an atom list, if the atom exists.`
- `long atom_arg_getdouble (double *c, long idx, long ac, const t_atom *av)`  
`Retrieves the floating point value, as a double, of a particular t_atom from an atom list, if the atom exists.`
- `long atom_arg_getsym (t_symbol **c, long idx, long ac, const t_atom *av)`  
`Retrieves the t_symbol* value of a particular t_atom from an atom list, if the atom exists.`
- `t_max_err atom_alloc (long *ac, t_atom **av, char *alloc)`  
`Allocate a single atom.`
- `t_max_err atom_alloc_array (long minsize, long *ac, t_atom **av, char *alloc)`  
`Allocate an array of atoms.`
- `t_max_err atom_setchar_array (long ac, t_atom *av, long count, unsigned char *vals)`  
`Assign an array of char values to an array of atoms.`
- `t_max_err atom_setlong_array (long ac, t_atom *av, long count, t_atom_long *vals)`  
`Assign an array of long integer values to an array of atoms.`
- `t_max_err atom_setfloat_array (long ac, t_atom *av, long count, float *vals)`  
`Assign an array of 32bit float values to an array of atoms.`
- `t_max_err atom_setdouble_array (long ac, t_atom *av, long count, double *vals)`  
`Assign an array of 64bit float values to an array of atoms.`
- `t_max_err atom_setsym_array (long ac, t_atom *av, long count, t_symbol **vals)`  
`Assign an array of t_symbol* values to an array of atoms.`
- `t_max_err atom_setatom_array (long ac, t_atom *av, long count, t_atom *vals)`  
`Assign an array of t_atom values to an array of atoms.`
- `t_max_err atom_setobj_array (long ac, t_atom *av, long count, t_object **vals)`

- **`t_max_err atom_setparse`** (long \*ac, `t_atom` \*\*av, C74\_CONST char \*parsestr)
 

*Assign an array of `t_object`\* values to an array of atoms.*
- **`t_max_err atom_setformat`** (long \*ac, `t_atom` \*\*av, C74\_CONST char \*fmt,...)
 

*Create an array of atoms populated with values using sprintf-like syntax.*
- **`t_max_err atom_getformat`** (long ac, `t_atom` \*av, C74\_CONST char \*fmt,...)
 

*Retrieve values from an array of atoms using sscanf-like syntax.*
- **`t_max_err atom_gettext`** (long ac, `t_atom` \*av, long \*textsize, char \*\*text, long flags)
 

*Convert an array of atoms into a C-string.*
- **`t_max_err atom_gettext_precision`** (long ac, `t_atom` \*av, long \*textsize, char \*\*text, long flags, long precision)
 

*Convert an array of atoms into a C-string, specifying floating-point precision.*
- **`t_max_err atom_getchar_array`** (long ac, `t_atom` \*av, long count, unsigned char \*vals)
 

*Fetch an array of char values from an array of atoms.*
- **`t_max_err atom_getlong_array`** (long ac, `t_atom` \*av, long count, `t_atom_long` \*vals)
 

*Fetch an array of long integer values from an array of atoms.*
- **`t_max_err atom_getfloat_array`** (long ac, `t_atom` \*av, long count, float \*vals)
 

*Fetch an array of 32bit float values from an array of atoms.*
- **`t_max_err atom_getdouble_array`** (long ac, `t_atom` \*av, long count, double \*vals)
 

*Fetch an array of 64bit float values from an array of atoms.*
- **`t_max_err atom_getsym_array`** (long ac, `t_atom` \*av, long count, `t_symbol` \*\*vals)
 

*Fetch an array of `t_symbol`\* values from an array of atoms.*
- **`t_max_err atom_getatom_array`** (long ac, `t_atom` \*av, long count, `t_atom` \*vals)
 

*Fetch an array of `t_atom` values from an array of atoms.*
- **`t_max_err atom_getobj_array`** (long ac, `t_atom` \*av, long count, `t_object` \*\*vals)
 

*Fetch an array of `t_object`\* values from an array of atoms.*
- **long `atomisstring`** (const `t_atom` \*a)

*Determines whether or not an atom represents a `t_string` object.*

- **long `atomisatomarray`** (`t_atom` \*a)

*Determines whether or not an atom represents a `t_atomarray` object.*

- **long `atomisdictionary`** (`t_atom` \*a)

*Determines whether or not an atom represents a `t_dictionary` object.*

- **BEGIN\_USING\_C\_LINKAGE void `atom_copy`** (long argc1, `t_atom` \*argv1, `t_atom` \*argv2)
 

*Copy an array of atoms.*
- **void `postargs`** (long argc, `t_atom` \*argv)
 

*Print the contents of an array of atoms to the Max window.*
- **`t_max_err atom_arg_getobjclass`** (`t_object` \*\*x, long idx, long argc, `t_atom` \*argv, `t_symbol` \*cls)
 

*Return a pointer to an object contained in an atom if it is of the specified class.*
- **void \* `atom_getobjclass`** (`t_atom` \*av, `t_symbol` \*cls)
 

*Return a pointer to an object contained in an atom if it is of the specified class.*

### 38.17.1 Detailed Description

### 38.17.2 Enumeration Type Documentation

### 38.17.2.1 e\_max\_atom\_gettext\_flags

enum [e\\_max\\_atom\\_gettext\\_flags](#)

Flags that determine how functions convert atoms into text (C-strings).

Enumerator

<a href="#">OBEX_UTIL_ATOM_GETTEXT_DEFAULT</a>	default translation rules for getting text from atoms
<a href="#">OBEX_UTIL_ATOM_GETTEXT_TRUNCATE_ZEROS</a>	eliminate redundant zeros for floating point numbers (default used)
<a href="#">OBEX_UTIL_ATOM_GETTEXT_SYM_NO_QUOTE</a>	don't introduce quotes around symbols with spaces
<a href="#">OBEX_UTIL_ATOM_GETTEXT_SYM_FORCE_QUOTE</a>	always introduce quotes around symbols (useful for JSON)
<a href="#">OBEX_UTIL_ATOM_GETTEXT_COMMAS_DELIM</a>	separate atoms with commas (useful for JSON)
<a href="#">OBEX_UTIL_ATOM_GETTEXT_FORCE_ZEROS</a>	always print the zeros
<a href="#">OBEX_UTIL_ATOM_GETTEXT_NUM_HI_RES</a>	print more decimal places
<a href="#">OBEX_UTIL_ATOM_GETTEXT_NUM_LO_RES</a>	print fewer decimal places (HI_RES will win though)
<a href="#">OBEX_UTIL_ATOM_GETTEXT_NOESCAPE</a>	don't add extra escape characters
<a href="#">OBEX_UTIL_ATOM_GETTEXT_LINEBREAK_↔_NODELIM</a>	don't insert spaces before/after linebreak characters

### 38.17.2.2 e\_max\_atomtypes

enum [e\\_max\\_atomtypes](#)

the list of officially recognized types, including pseudotypes for commas and semicolons.

Used in two places: 1. the reader, when it reads a string, returns long, float, sym, comma, semi, or dollar; and 2. each object method comes with an array of them saying what types it needs, from among long, float, sym, obj, gimme, and cant.

Remarks

While these values are defined in an enum, you should use a long to represent the value. Using the enum type creates ambiguity in struct size and is subject to various inconsistent compiler settings.

Enumerator

<a href="#">A NOTHING</a>	no type, thus no atom
<a href="#">A LONG</a>	long integer
<a href="#">A FLOAT</a>	32-bit float
<a href="#">A SYM</a>	<a href="#">t_symbol</a> pointer
<a href="#">A OBJ</a>	<a href="#">t_object</a> pointer (for argtype lists; passes the value of sym)
<a href="#">A DEFLONG</a>	long but defaults to zero

## Enumerator

A_DEFFLOAT	float, but defaults to zero
A_DEFSYM	symbol, defaults to ""
A_GIMME	request that args be passed as an array, the routine will check the types itself.
A_CANT	cannot typecheck args
A_SEMI	semicolon
A_COMMA	comma
A_DOLLAR	dollar
A_DOLLSYM	dollar
A_GIMMEBACK	request that args be passed as an array, the routine will check the types itself. can return atom value in final atom ptr arg. function returns long error code 0 = no err. see gimmeback_meth typedef
A_DEFER	A special signature for declaring methods. This is like A_GIMME, but the call is deferred.
A_USURP	A special signature for declaring methods. This is like A_GIMME, but the call is deferred and multiple calls within one servicing of the queue are filtered down to one call.
A_DEFER_LOW	A special signature for declaring methods. This is like A_GIMME, but the call is deferref to the back of the queue.
A_USURP_LOW	A special signature for declaring methods. This is like A_GIMME, but the call is deferred to the back of the queue and multiple calls within one servicing of the queue are filtered down to one call.

## 38.17.3 Function Documentation

## 38.17.3.1 atom\_alloc()

```
t_max_err atom_alloc (
    long * ac,
    t_atom ** av,
    char * alloc )
```

Allocate a single atom.

If ac and av are both zero then memory is allocated. Otherwise it is presumed that memory is already allocated and nothing will happen.

## Parameters

ac	The address of a variable that will contain the number of atoms allocated (1).
av	The address of a pointer that will be set with the new allocated memory for the atom.
alloc	Address of a variable that will be set true if memory is allocated, otherwise false.

**Returns**

A Max error code.

**38.17.3.2 atom\_alloc\_array()**

```
t_max_err atom_alloc_array (
    long minsize,
    long * ac,
    t_atom ** av,
    char * alloc )
```

Allocate an array of atoms.

If ac and av are both zero then memory is allocated. Otherwise it is presumed that memory is already allocated and nothing will happen.

**Parameters**

<i>minsize</i>	The minimum number of atoms that this array will need to contain. This determines the amount of memory allocated.
<i>ac</i>	The address of a variable that will contain the number of atoms allocated.
<i>av</i>	The address of a pointer that will be set with the new allocated memory for the atoms.
<i>alloc</i>	Address of a variable that will be set true if memory is allocated, otherwise false.

**Returns**

A Max error code.

**38.17.3.3 atom\_arg\_getdouble()**

```
long atom_arg_getdouble (
    double * c,
    long idx,
    long ac,
    const t_atom * av )
```

Retrieves the floating point value, as a double, of a particular [t\\_atom](#) from an atom list, if the atom exists.

**Parameters**

<i>c</i>	Pointer to a double variable to receive the atom's data if the function is successful. Otherwise the value is left unchanged.
<i>idx</i>	Offset into the atom list of the atom of interest, starting from 0. For instance, if you want data from the 3rd atom in the atom list, <i>idx</i> should be set to 2.
<i>ac</i>	Count of av.
<i>av</i>	Pointer to the first <a href="#">t_atom</a> of an atom list.

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `jit_atom_arg_getdouble()`.

**38.17.3.4 atom\_arg\_getfloat()**

```
long atom_arg_getfloat (
    float * c,
    long idx,
    long ac,
    const t_atom * av )
```

Retrieves the floating point value of a particular `t_atom` from an atom list, if the atom exists.

**Parameters**

<code>c</code>	Pointer to a float variable to receive the atom's data if the function is successful. Otherwise, the value is left unchanged.
<code>idx</code>	Offset into the atom list of the atom of interest, starting from 0. For instance, if you want data from the 3rd atom in the atom list, <code>idx</code> should be set to 2.
<code>ac</code>	Count of <code>av</code> .
<code>av</code>	Pointer to the first <code>t_atom</code> of an atom list.

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `jit_atom_arg_getfloat()`.

**38.17.3.5 atom\_arg\_getlong()**

```
t_max_err atom_arg_getlong (
    t_atom_long * c,
    long idx,
    long ac,
    const t_atom * av )
```

Retrieves the integer value of a particular `t_atom` from an atom list, if the atom exists.

**Parameters**

<i>c</i>	Pointer to a long variable to receive the atom's data if the function is successful.
<i>idx</i>	Offset into the atom list of the atom of interest, starting from 0. For instance, if you want data from the 3rd atom in the atom list, <i>idx</i> should be set to 2.
<i>ac</i>	Count of av.
<i>av</i>	Pointer to the first <a href="#">t_atom</a> of an atom list.

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

**Remarks**

The [atom\\_arg\\_getlong\(\)](#) function only changes the value of *c* if the function is successful. For instance, the following code snippet illustrates a simple, but typical use:

```
void myobject_mymessage(t_myobject **x, t_symbol *s, long ac, t_atom *av)
{
    t_atom_long var = -1;
    // here, we are expecting a value of 0 or greater
    atom_arg_getlong(&var, 0, ac, av);
    if (val == -1) // i.e. unchanged
        post("it is likely that the user did not provide a valid argument");
    else {
        ...
    }
}
```

Referenced by [jit\\_atom\\_arg\\_getlong\(\)](#).

**38.17.3.6 atom\_arg\_getobjclass()**

```
t_max_err atom_arg_getobjclass (
    t_object ** x,
    long idx,
    long argc,
    t_atom * argv,
    t_symbol * cls )
```

Return a pointer to an object contained in an atom if it is of the specified class.

**Parameters**

<i>x</i>	The address of a pointer to the object contained in <i>av</i> if it is of the specified class upon return. Otherwise NULL upon return.
<i>idx</i>	The index of the atom in the array from which to get the object pointer.
<i>argc</i>	The count of atoms in <i>argv</i> .
<i>argv</i>	The address to the first of an array of atoms.
<i>cls</i>	A symbol containing the class name of which the object should be an instance.

**Returns**

A Max error code.

**38.17.3.7 atom\_arg\_getsym()**

```
long atom_arg_getsym (
    t_symbol ** c,
    long idx,
    long ac,
    const t_atom * av )
```

Retrieves the `t_symbol` \* value of a particular `t_atom` from an atom list, if the atom exists.

**Parameters**

<code>c</code>	Pointer to a <code>t_symbol</code> * variable to receive the atom's data if the function is successful. Otherwise, the value is left unchanged.
<code>idx</code>	Offset into the atom list of the atom of interest, starting from 0. For instance, if you want data from the 3rd atom in the atom list, <code>idx</code> should be set to 2.
<code>ac</code>	Count of av.
<code>av</code>	Pointer to the first <code>t_atom</code> of an atom list.

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

**Remarks**

The `atom_arg_getsym()` function only changes the value of `c` if the function is successful. For instance, the following code snippet illustrates a simple, but typical use:

```
void myobject_open(t_myobject **x, t_symbol *s, long ac, t_atom *av)
{
    t_symbol *filename = _sym_nothing;
    // here, we are expecting a file name.
    // if we don't get it, open a dialog box
    atom_arg_getsym(&filename, 0, ac, av);
    if (filename == _sym_nothing) { // i.e. unchanged
        // open the file dialog box,
        // get a value for filename
    }
    // do something with the filename
}
```

Referenced by `jit_atom_arg_getsym()`.

### 38.17.3.8 atom\_copy()

```
BEGIN_USING_C_LINKAGE void atom_copy (
    long argc1,
    t_atom * argv1,
    t_atom * argv2 )
```

Copy an array of atoms.

#### Parameters

<i>argc1</i>	The count of atoms in argv1.
<i>argv1</i>	The address to the first of an array of atoms that is the source for the copy.
<i>argv2</i>	The address to the first of an array of atoms that is the destination for the copy. Note that this array must already be allocated using <a href="#">sysmem_newptr()</a> or <a href="#">atom_alloc()</a> .

### 38.17.3.9 atom\_getatom\_array()

```
t_max_err atom_getatom_array (
    long ac,
    t_atom * av,
    long count,
    t_atom * vals )
```

Fetch an array of [t\\_atom](#) values from an array of atoms.

#### Parameters

<i>ac</i>	The number of atoms allocated in the <i>av</i> parameter.
<i>av</i>	The address to the first of an array of allocated atoms.
<i>count</i>	The number of values to fetch from the array specified by <i>vals</i> .
<i>vals</i>	The address of the array to which is copied the values from <i>av</i> .

#### Returns

A Max error code.

### 38.17.3.10 atom\_getchar\_array()

```
t_max_err atom_getchar_array (
    long ac,
```

```
t_atom * av,
long count,
unsigned char * vals )
```

Fetch an array of char values from an array of atoms.

#### Parameters

<i>ac</i>	The number of atoms allocated in the <i>av</i> parameter.
<i>av</i>	The address to the first of an array of allocated atoms.
<i>count</i>	The number of values to fetch from the array specified by <i>vals</i> .
<i>vals</i>	The address of the array to which is copied the values from <i>av</i> .

#### Returns

A Max error code.

### 38.17.3.11 atom\_getcharfix()

```
long atom_getcharfix (
    const t_atom * a )
```

Retrieves an unsigned integer value between 0 and 255 from a [t\\_atom](#).

#### Parameters

<i>a</i>	Pointer to a <a href="#">t_atom</a> whose value is of interest
----------	--

#### Returns

This function returns the value of the specified [t\\_atom](#) as an integer between 0 and 255, if possible. Otherwise, it returns 0.

#### Remarks

If the [t\\_atom](#) is typed [A\\_LONG](#), but the data falls outside of the range 0-255, the data is truncated to that range before output.

If the [t\\_atom](#) is typed [A\\_FLOAT](#), the floating point value is multiplied by 255. and truncated to the range 0-255 before output. For example, the floating point value 0 . 5 would be output from `atom_getcharfix` as 127 ( $0.5 * 255 = 127.5$ ).

No attempt is also made to coerce [t\\_symbol](#) data.

Referenced by `jit_atom_getcharfix()`.

### 38.17.3.12 atom\_getdouble\_array()

```
t_max_err atom_getdouble_array (
    long ac,
    t_atom * av,
    long count,
    double * vals )
```

Fetch an array of 64bit float values from an array of atoms.

#### Parameters

<i>ac</i>	The number of atoms allocated in the <i>av</i> parameter.
<i>av</i>	The address to the first of an array of allocated atoms.
<i>count</i>	The number of values to fetch from the array specified by <i>vals</i> .
<i>vals</i>	The address of the array to which is copied the values from <i>av</i> .

#### Returns

A Max error code.

### 38.17.3.13 atom\_getfloat()

```
t_atom_float atom_getfloat (
    const t_atom * a )
```

Retrieves a floating point value from a [t\\_atom](#).

#### Parameters

<i>a</i>	Pointer to a <a href="#">t_atom</a> whose value is of interest
----------	--

#### Returns

This function returns the value of the specified [t\\_atom](#) as a floating point number, if possible. Otherwise, it returns 0.

#### Remarks

If the [t\\_atom](#) is not of the type specified by the function, the function will attempt to coerce a valid value from the [t\\_atom](#). For instance, if the [t\\_atom](#) *at* is set to type [A\\_LONG](#) with a value of 5, the [atom\\_getfloat\(\)](#) function will return the value of *at* as a float, or 5.0. An attempt is also made to coerce [t\\_symbol](#) data.

Referenced by [jit\\_atom\\_getfloat\(\)](#).

### 38.17.3.14 atom\_getfloat\_array()

```
t_max_err atom_getfloat_array (
    long ac,
    t_atom * av,
    long count,
    float * vals )
```

Fetch an array of 32bit float values from an array of atoms.

#### Parameters

<i>ac</i>	The number of atoms allocated in the <i>av</i> parameter.
<i>av</i>	The address to the first of an array of allocated atoms.
<i>count</i>	The number of values to fetch from the array specified by <i>vals</i> .
<i>vals</i>	The address of the array to which is copied the values from <i>av</i> .

#### Returns

A Max error code.

### 38.17.3.15 atom\_getformat()

```
t_max_err atom_getformat (
    long ac,
    t_atom * av,
    C74_CONST char * fmt,
    ...
)
```

Retrieve values from an array of atoms using sscanf-like syntax.

*atom\_getformat()* supports clfdsoaCLFDOSA tokens (primitive type scalars and arrays respectively for the char, long, float, double, *t\_symbol*\*, *t\_object*\*, *t\_atom*\*). It does not support vbp@ the tokens found in *atom\_setformat()*.

#### Parameters

<i>ac</i>	The number of atoms to parse in <i>av</i> .
<i>av</i>	The address of the first <i>t_atom</i> pointer in an array to parse.
<i>fmt</i>	An sscanf-style format string specifying types for the atoms.
...	One or more arguments which are address of variables to be set according to the <i>fmt</i> string.

#### Returns

A Max error code.

## See also

[atom\\_setformat\(\)](#)

**38.17.3.16 atom\_getlong()**

```
t_atom_long atom_getlong (
    const t_atom * a )
```

Retrieves a long integer value from a [t\\_atom](#).

## Parameters

a	Pointer to a <a href="#">t_atom</a> whose value is of interest
---	--

## Returns

This function returns the value of the specified [t\\_atom](#) as an integer, if possible. Otherwise, it returns 0.

## Remarks

If the [t\\_atom](#) is not of the type specified by the function, the function will attempt to coerce a valid value from the [t\\_atom](#). For instance, if the [t\\_atom](#) at is set to type [A\\_FLOAT](#) with a value of 3.7, the [atom\\_getlong\(\)](#) function will return the truncated integer value of at, or 3. An attempt is also made to coerce [t\\_symbol](#) data.

Referenced by [jit\\_atom\\_getlong\(\)](#).

**38.17.3.17 atom\_getlong\_array()**

```
t_max_err atom_getlong_array (
    long ac,
    t_atom * av,
    long count,
    t_atom_long * vals )
```

Fetch an array of long integer values from an array of atoms.

## Parameters

ac	The number of atoms allocated in the av parameter.
av	The address to the first of an array of allocated atoms.
count	The number of values to fetch from the array specified by vals.
vals	The address of the array to which is copied the values from av.

**Returns**

A Max error code.

**38.17.3.18 atom\_getobj()**

```
void* atom_getobj (
    const t_atom * a )
```

Retrieves a generic pointer value from a [t\\_atom](#).

**Parameters**

<i>a</i>	Pointer to a <a href="#">t_atom</a> whose value is of interest
----------	--

**Returns**

This function returns the value of the specified [A\\_OBJ](#)-typed [t\\_atom](#), if possible. Otherwise, it returns NULL.

Referenced by [jit\\_atom\\_getobj\(\)](#).

**38.17.3.19 atom\_getobj\_array()**

```
t_max_err atom_getobj_array (
    long ac,
    t_atom * av,
    long count,
    t_object ** vals )
```

Fetch an array of [t\\_object](#)\* values from an array of atoms.

**Parameters**

<i>ac</i>	The number of atoms allocated in the <i>av</i> parameter.
<i>av</i>	The address to the first of an array of allocated atoms.
<i>count</i>	The number of values to fetch from the array specified by <i>vals</i> .
<i>vals</i>	The address of the array to which is copied the values from <i>av</i> .

**Returns**

A Max error code.

### 38.17.3.20 atom\_getobjclass()

```
void* atom_getobjclass (
    t_atom * av,
    t_symbol * cls )
```

Return a pointer to an object contained in an atom if it is of the specified class.

#### Parameters

<i>av</i>	A pointer to the atom from which to get the <a href="#">t_object</a> .
<i>cls</i>	A symbol containing the class name of which the object should be an instance.

#### Returns

A pointer to the object contained in *av* if it is of the specified class, otherwise NULL.

### 38.17.3.21 atom\_getsym()

```
t_symbol* atom_getsym (
    const t_atom * a )
```

Retrieves a [t\\_symbol](#) \* value from a [t\\_atom](#).

#### Parameters

<i>a</i>	Pointer to a <a href="#">t_atom</a> whose value is of interest
----------	--

#### Returns

This function returns the value of the specified [A\\_SYM](#)-typed [t\\_atom](#), if possible. Otherwise, it returns an empty, but valid, [t\\_symbol](#) \*, equivalent to `gensym("")`, or `_sym_nothing`.

#### Remarks

No attempt is made to coerce non-matching data types.

Referenced by `jit_atom_getsym()`.

### 38.17.3.22 atom\_getsym\_array()

```
t_max_err atom_getsym_array (
    long ac,
    t_atom * av,
    long count,
    t_symbol ** vals )
```

Fetch an array of `t_symbol`\* values from an array of atoms.

#### Parameters

<code>ac</code>	The number of atoms allocated in the <code>av</code> parameter.
<code>av</code>	The address to the first of an array of allocated atoms.
<code>count</code>	The number of values to fetch from the array specified by <code>vals</code> .
<code>vals</code>	The address of the array to which is copied the values from <code>av</code> .

#### Returns

A Max error code.

### 38.17.3.23 atom\_gettext()

```
t_max_err atom_gettext (
    long ac,
    t_atom * av,
    long * textsize,
    char ** text,
    long flags )
```

Convert an array of atoms into a C-string.

#### Parameters

<code>ac</code>	The number of atoms to fetch in <code>av</code> .
<code>av</code>	The address of the first <code>t_atom</code> pointer in an array to retrieve.
<code>textsize</code>	The size of the string to which the atoms will be formatted and copied.
<code>text</code>	The address of the string to which the text will be written.
<code>flags</code>	Determines the rules by which atoms will be translated into text. Values are bit mask as defined by <code>e_max_atom_gettext_flags</code> .

#### Returns

A Max error code.

## See also

[atom\\_setparse\(\)](#)

**38.17.3.24 atom\_gettext\_precision()**

```
t_max_err atom_gettext_precision (
    long ac,
    t_atom * av,
    long * textsize,
    char ** text,
    long flags,
    long precision )
```

Convert an array of atoms into a C-string, specifying floating-point precision.

## Parameters

<i>ac</i>	The number of atoms to fetch in <i>av</i> .
<i>av</i>	The address of the first <a href="#">t_atom</a> pointer in an array to retrieve.
<i>textsize</i>	The size of the string to which the atoms will be formatted and copied.
<i>text</i>	The address of the string to which the text will be written.
<i>flags</i>	Determines the rules by which atoms will be translated into text. Values are bit mask as defined by <a href="#">e_max_atom_gettext_flags</a> .
<i>precision</i>	Determines the number of digits after the decimal point when floats are translated into text. This overrides the OBEX_UTIL_ATOM_GETTEXT_NUM_HI_RES and OBEX_UTIL_ATOM_GETTEXT_NUM_LO_RES <a href="#">e_max_atom_gettext_flags</a> . Pass -1 to ignore this value and behave as <a href="#">atom_gettext()</a> .

## Returns

A Max error code.

## See also

[atom\\_setparse\(\)](#)

**38.17.3.25 atom\_gettype()**

```
long atom_gettype (
    const t_atom * a )
```

Retrieves type from a [t\\_atom](#).

**Parameters**

<i>a</i>	Pointer to a <a href="#">t_atom</a> whose type is of interest
----------	---

**Returns**

This function returns the type of the specified [t\\_atom](#) as defined in [e\\_max\\_atomtypes](#)

**38.17.3.26 atom\_setatom\_array()**

```
t_max_err atom_setatom_array (
    long ac,
    t_atom * av,
    long count,
    t_atom * vals )
```

Assign an array of [t\\_atom](#) values to an array of atoms.

**Parameters**

<i>ac</i>	The number of atoms to try to fetch from the second array of atoms. You should have at least this number of atoms allocated in <i>av</i> .
<i>av</i>	The address to the first of an array of allocated atoms.
<i>count</i>	The number of values in the array specified by <i>vals</i> .
<i>vals</i>	The array from which to copy the values into the array of atoms at <i>av</i> .

**Returns**

A Max error code.

**38.17.3.27 atom\_setchar\_array()**

```
t_max_err atom_setchar_array (
    long ac,
    t_atom * av,
    long count,
    unsigned char * vals )
```

Assign an array of char values to an array of atoms.

**Parameters**

<i>ac</i>	The number of atoms to try to fetch from the array of chars. You should have at least this number of atoms allocated in av.
<i>av</i>	The address to the first of an array of allocated atoms.
<i>count</i>	The number of values in the array specified by vals.
<i>vals</i>	The array from which to copy the values into the array of atoms at av.

**Returns**

A Max error code.

**38.17.3.28 atom\_setdouble\_array()**

```
t_max_err atom_setdouble_array (
    long ac,
    t_atom * av,
    long count,
    double * vals )
```

Assign an array of 64bit float values to an array of atoms.

**Parameters**

<i>ac</i>	The number of atoms to try to fetch from the array of doubles. You should have at least this number of atoms allocated in av.
<i>av</i>	The address to the first of an array of allocated atoms.
<i>count</i>	The number of values in the array specified by vals.
<i>vals</i>	The array from which to copy the values into the array of atoms at av.

**Returns**

A Max error code.

**38.17.3.29 atom\_setfloat()**

```
t_max_err atom_setfloat (
    t_atom * a,
    double b )
```

Inserts a floating point number into a [t\\_atom](#) and change the [t\\_atom](#)'s type to [A\\_FLOAT](#).

**Parameters**

<i>a</i>	Pointer to a <a href="#">t_atom</a> whose value and type will be modified
<i>b</i>	Floating point value to copy into the <a href="#">t_atom</a>

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

Referenced by [jit\\_atom\\_setfloat\(\)](#).

**38.17.3.30 atom\_setfloat\_array()**

```
t_max_err atom_setfloat_array (
    long ac,
    t_atom * av,
    long count,
    float * vals )
```

Assign an array of 32bit float values to an array of atoms.

**Parameters**

<i>ac</i>	The number of atoms to try to fetch from the array of floats. You should have at least this number of atoms allocated in <i>av</i> .
<i>av</i>	The address to the first of an array of allocated atoms.
<i>count</i>	The number of values in the array specified by <i>vals</i> .
<i>vals</i>	The array from which to copy the values into the array of atoms at <i>av</i> .

**Returns**

A Max error code.

**38.17.3.31 atom\_setformat()**

```
t_max_err atom_setformat (
    long * ac,
    t_atom ** av,
    C74_CONST char * fmt,
    ... )
```

Create an array of atoms populated with values using sprintf-like syntax.

[atom\\_setformat\(\)](#) supports clfdsoaCLFDOSA tokens (primitive type scalars and arrays respectively for the char, long, float, double, [t\\_symbol\\*](#), [t\\_object\\*](#), [t\\_atom\\*](#)). It also supports vbp@ tokens (obval, binbuf, parsestr, attribute).

This function allocates memory for the atoms if the ac and av parameters are NULL. Otherwise it will attempt to use any memory already allocated to av. Any allocated memory should be freed with [sysmem\\_freeptr\(\)](#).

#### Parameters

<i>ac</i>	The address of a variable to hold the number of returned atoms.
<i>av</i>	The address of a <a href="#">t_atom</a> pointer to which memory may be allocated and atoms copied.
<i>fmt</i>	An sprintf-style format string specifying values for the atoms.
...	One or more arguments which are to be substituted into the format string.

#### Returns

A Max error code.

#### See also

[atom\\_getformat\(\)](#)  
[atom\\_setparse\(\)](#)

### 38.17.3.32 atom\_setlong()

```
t_max_err atom_setlong (
    t_atom * a,
    t_atom_long b )
```

Inserts an integer into a [t\\_atom](#) and change the [t\\_atom](#)'s type to [A\\_LONG](#).

#### Parameters

<i>a</i>	Pointer to a <a href="#">t_atom</a> whose value and type will be modified
<i>b</i>	Integer value to copy into the <a href="#">t_atom</a>

#### Returns

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

Referenced by [jit\\_atom\\_setlong\(\)](#).

### 38.17.3.33 atom\_setlong\_array()

```
t_max_err atom_setlong_array (
    long ac,
    t_atom * av,
    long count,
    t_atom_long * vals )
```

Assign an array of long integer values to an array of atoms.

#### Parameters

<i>ac</i>	The number of atoms to try to fetch from the array of longs. You should have at least this number of atoms allocated in <i>av</i> .
<i>av</i>	The address to the first of an array of allocated atoms.
<i>count</i>	The number of values in the array specified by <i>vals</i> .
<i>vals</i>	The array from which to copy the values into the array of atoms at <i>av</i> .

#### Returns

A Max error code.

### 38.17.3.34 atom\_setobj()

```
t_max_err atom_setobj (
    t_atom * a,
    void * b )
```

Inserts a generic pointer value into a *t\_atom* and change the *t\_atom*'s type to *A\_OBJ*.

#### Parameters

<i>a</i>	Pointer to a <i>t_atom</i> whose value and type will be modified
<i>b</i>	Pointer value to copy into the <i>t_atom</i>

#### Returns

This function returns the error code *MAX\_ERR\_NONE* if successful, or one of the other error codes defined in *e\_max\_errorcodes* if unsuccessful.

Referenced by *jit\_atom\_setobj()*.

### 38.17.3.35 atom\_setobj\_array()

```
t_max_err atom_setobj_array (
    long ac,
    t_atom * av,
    long count,
    t_object ** vals )
```

Assign an array of [t\\_object](#)\* values to an array of atoms.

#### Parameters

<i>ac</i>	The number of atoms to try to fetch from the array of objects. You should have at least this number of atoms allocated in <i>av</i> .
<i>av</i>	The address to the first of an array of allocated atoms.
<i>count</i>	The number of values in the array specified by <i>vals</i> .
<i>vals</i>	The array from which to copy the values into the array of atoms at <i>av</i> .

#### Returns

A Max error code.

### 38.17.3.36 atom\_setparse()

```
t_max_err atom_setparse (
    long * ac,
    t_atom ** av,
    C74_CONST char * parsestr )
```

Parse a C-string into an array of atoms.

This function allocates memory for the atoms if the *ac* and *av* parameters are NULL. Otherwise it will attempt to use any memory already allocated to *av*. Any allocated memory should be freed with [sysmem\\_freeptr\(\)](#).

#### Parameters

<i>ac</i>	The address of a variable to hold the number of returned atoms.
<i>av</i>	The address of a <a href="#">t_atom</a> pointer to which memory may be allocated and atoms copied.
<i>parsestr</i>	The C-string to parse.

#### Returns

A Max error code.

### Remarks

The following example will parse the string "foo bar 1 2 3.0" into an array of 5 atoms. The atom types will be determined automatically as 2 `A_SYM` atoms, 2 `A_LONG` atoms, and 1 `A_FLOAT` atom.

```
t_atom *av = NULL;
long ac = 0;
t_max_err err = MAX_ERR_NONE;
err = atom_setparse(&ac, &av, "foo bar 1 2 3.0");
```

### 38.17.3.37 atom\_setsym()

```
t_max_err atom_setsym (
    t_atom * a,
    t_symbol * b )
```

Inserts a `t_symbol`\* into a `t_atom` and change the `t_atom`'s type to `A_SYM`.

#### Parameters

<code>a</code>	Pointer to a <code>t_atom</code> whose value and type will be modified
<code>b</code>	Pointer to a <code>t_symbol</code> to copy into the <code>t_atom</code>

#### Returns

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `jit_atom_setsym()`.

### 38.17.3.38 atom\_setsym\_array()

```
t_max_err atom_setsym_array (
    long ac,
    t_atom * av,
    long count,
    t_symbol ** vals )
```

Assign an array of `t_symbol`\* values to an array of atoms.

#### Parameters

<code>ac</code>	The number of atoms to try to fetch from the array of symbols. You should have at least this number of atoms allocated in <code>av</code> .
<code>av</code>	The address to the first of an array of allocated atoms.
<code>count</code>	The number of values in the array specified by <code>vals</code> .
<code>vals</code>	The array from which to copy the values into the array of atoms at <code>av</code> .

**Returns**

A Max error code.

**38.17.3.39 atomisatomarray()**

```
long atomisatomarray (
    t_atom * a )
```

Determines whether or not an atom represents a [t\\_atomarray](#) object.

**Parameters**

a	The address of the atom to test.
---	----------------------------------

**Returns**

Returns true if the [t\\_atom](#) contains a valid [t\\_atomarray](#) object.

**38.17.3.40 atomisdictionary()**

```
long atomisdictionary (
    t_atom * a )
```

Determines whether or not an atom represents a [t\\_dictionary](#) object.

**Parameters**

a	The address of the atom to test.
---	----------------------------------

**Returns**

Returns true if the [t\\_atom](#) contains a valid [t\\_dictionary](#) object.

**38.17.3.41 atomisstring()**

```
long atomisstring (
    const t_atom * a )
```

Determines whether or not an atom represents a [t\\_string](#) object.

**Parameters**

<i>a</i>	The address of the atom to test.
----------	----------------------------------

**Returns**

Returns true if the [t\\_atom](#) contains a valid [t\\_string](#) object.

**38.17.3.42 postargs()**

```
void postargs (
    long argc,
    t_atom * argv )
```

Print the contents of an array of atoms to the Max window.

**Parameters**

<i>argc</i>	The count of atoms in argv.
<i>argv</i>	The address to the first of an array of atoms.

**38.18 Atombufs**

An Atombuf is an alternative to [Binbufs](#) for temporary storage of atoms.

Collaboration diagram for Atombufs:

**Data Structures**

- struct [t\\_atombuf](#)

*The atombuf struct provides a way to pass a collection of atoms.*

## Functions

- **BEGIN\_USING\_C\_LINKAGE void \* atombuf\_new (long argc, t\_atom \*argv)**  
*Use [atombuf\\_new\(\)](#) to create a new Atombuf from an array of t\_atoms.*
- **void atombuf\_free (t\_atombuf \*x)**  
*Use [atombuf\\_free\(\)](#) to dispose of the memory used by a t\_atombuf.*
- **void atombuf\_text (t\_atombuf \*\*x, char \*\*text, long size)**  
*Use [atombuf\\_text\(\)](#) to convert text to a t\_atom array in a t\_atombuf.*

### 38.18.1 Detailed Description

An Atombuf is an alternative to [Binbufs](#) for temporary storage of atoms.

Its principal advantage is that the internal structure is publicly available so you can manipulate the atoms in place. The standard Max text objects (message box, object box, comment) use the Atombuf structure to store their text (each word of text is stored as a [t\\_symbol](#) or a number).

### 38.18.2 Function Documentation

#### 38.18.2.1 atombuf\_free()

```
void atombuf_free (
    t_atombuf * x )
```

Use [atombuf\\_free\(\)](#) to dispose of the memory used by a t\_atombuf.

##### Parameters

x	The t_atombuf to free.
---	------------------------

#### 38.18.2.2 atombuf\_new()

```
BEGIN_USING_C_LINKAGE void* atombuf_new (
    long argc,
    t_atom * argv )
```

Use [atombuf\\_new\(\)](#) to create a new Atombuf from an array of t\_atoms.

**Parameters**

<i>argc</i>	Number of t_atoms in the argv array. May be 0.
<i>argv</i>	Array of t_atoms. If creating an empty Atombuf, you may pass 0.

**Returns**

[atombuf\\_new\(\)](#) create a new t\_atombuf and returns a pointer to it. If 0 is returned, insufficient memory was available.

**38.18.2.3 atombuf\_text()**

```
void atombuf_text (
    t_atombuf ** x,
    char ** text,
    long size )
```

Use [atombuf\\_text\(\)](#) to convert text to a t\_atom array in a t\_atombuf.

To use this routine to create a new Atombuf from the text buffer, first create a new empty t\_atombuf with a call to `atombuf_new(0,NULL)`.

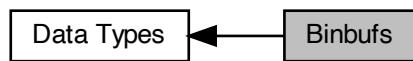
**Parameters**

<i>x</i>	Pointer to existing atombuf variable. The variable will be replaced by a new Atombuf containing the converted text.
<i>text</i>	Handle to the text to be converted. It need not be zero-terminated.
<i>size</i>	Number of characters in the text.

**38.19 Binbufs**

You won't need to know about the internal structure of a Binbuf, so you can use the void \* type to refer to one.

Collaboration diagram for Binbufs:



## Functions

- void \* **binbuf\_new** (void)
 

*Use binbuf\_new() to create and initialize a Binbuf.*
- void **binbuf\_vinsert** (void \*x, char \*fmt,...)
 

*Use binbuf\_vinsert() to append a Max message to a Binbuf adding a semicolon.*
- void **binbuf\_insert** (t\_binbuf \*x, t\_symbol \*s, short argc, t\_atom \*argv)
 

*Use binbuf\_insert() to append a Max message to a Binbuf adding a semicolon.*
- void \* **binbuf\_eval** (t\_binbuf \*x, short ac, t\_atom \*av, void \*to)
 

*Use binbuf\_eval to evaluate a Max message in a Binbuf, passing it arguments.*
- short **binbuf\_getatom** (t\_binbuf \*x, long \*p1, long \*p2, t\_atom \*ap)
 

*Use binbuf\_getatom to retrieve a single t\_atom from a Binbuf.*
- short **binbuf\_text** (t\_binbuf \*x, char \*\*srcText, long n)
 

*Use binbuf\_text() to convert a text handle to a Binbuf.*
- short **binbuf\_totext** (t\_binbuf \*x, char \*\*dstText, t\_ptr\_size \*sizep)
 

*Use binbuf\_totext() to convert a Binbuf into a text handle.*
- void **binbuf\_set** (t\_binbuf \*x, t\_symbol \*s, short argc, t\_atom \*argv)
 

*Use binbuf\_set() to change the entire contents of a Binbuf.*
- void **binbuf\_append** (t\_binbuf \*x, t\_symbol \*s, short argc, t\_atom \*argv)
 

*Use binbuf\_append to append t\_atoms to a Binbuf without modifying them.*
- short **readatom** (char \*outstr, char \*\*text, long \*n, long e, t\_atom \*ap)
 

*Use readatom() to read a single t\_atom from a text buffer.*

### 38.19.1 Detailed Description

You won't need to know about the internal structure of a Binbuf, so you can use the void \* type to refer to one.

### 38.19.2 Function Documentation

#### 38.19.2.1 binbuf\_append()

```
void binbuf_append (
    t_binbuf * x,
    t_symbol * s,
    short argc,
    t_atom * argv )
```

Use binbuf\_append to append t\_atoms to a Binbuf without modifying them.

#### Parameters

x	Binbuf to receive the items.
s	Ignored. Pass NULL.
argc	Count of items in the argv array.
argv	Array of atoms to add to the Binbuf.

### 38.19.2.2 binbuf\_eval()

```
void* binbuf_eval (
    t_binbuf * x,
    short ac,
    t_atom * av,
    void * to )
```

Use binbuf\_eval to evaluate a Max message in a Binbuf, passing it arguments.

[binbuf\\_eval\(\)](#) is an advanced function that evaluates the message in a Binbuf with arguments in argv, and sends it to receiver.

#### Parameters

x	Binbuf containing the message.
ac	Count of items in the argv array.
av	Array of t_atoms as the arguments to the message.
to	Receiver of the message.

#### Returns

The result of sending the message.

### 38.19.2.3 binbuf\_getatom()

```
short binbuf_getatom (
    t_binbuf * x,
    long * p1,
    long * p2,
    t_atom * ap )
```

Use binbuf\_getatom to retrieve a single [t\\_atom](#) from a Binbuf.

#### Parameters

x	Binbuf containing the desired <a href="#">t_atom</a> .
p1	Offset into the Binbuf's array of types. Modified to point to the next <a href="#">t_atom</a> .
p2	Offset into the Binbuf's array of data. Modified to point to the next <a href="#">t_atom</a> .
ap	Location of a <a href="#">t_atom</a> where the retrieved data will be placed.

**Returns**

1 if there were no t\_atoms at the specified offsets, 0 if there's a legitimate t\_atom returned in result.

**Remarks**

To get the first t\_atom, set both typeOffset and stuffOffset to 0. Here's an example of getting all the items in a Binbuf:

```
t_atom holder;
long to, so;
to = 0;
so = 0;
while (!binbuf_getatom(x, &to, &so, &holder)){
    // do something with the t_atom
}
```

**38.19.2.4 binbuf\_insert()**

```
void binbuf_insert (
    t_binbuf * x,
    t_symbol * s,
    short argc,
    t_atom * argv )
```

Use [binbuf\\_insert\(\)](#) to append a Max message to a Binbuf adding a semicolon.

**Parameters**

<i>x</i>	Binbuf to receive the items.
<i>s</i>	Ignored. Pass NULL.
<i>argc</i>	Count of items in the argv array.
<i>argv</i>	Array of t_atoms to add to the Binbuf.

**Remarks**

You'll use [binbuf\\_insert\(\)](#) instead of [binbuf\\_append\(\)](#) if you were saving your object into a Binbuf and wanted a semicolon at the end. If the message is part of a file that will later be evaluated, such as a Patcher file, the first argument argv[0] will be the receiver of the message and must be a Symbol. [binbuf\\_vinsert\(\)](#) is easier to use than [binbuf\\_insert\(\)](#), since you don't have to format your data into an array of Atoms first.

[binbuf\\_insert\(\)](#) will also convert the t\_symbols #1 through #9 into \$1 through \$9. This is used for saving patcher files that take arguments; you will probably never save these symbols as part of anything you are doing.

**38.19.2.5 binbuf\_new()**

```
void* binbuf_new (
    void )
```

Use [binbuf\\_new\(\)](#) to create and initialize a Binbuf.

**Returns**

Returns a new binbuf if successful, otherwise NULL.

**38.19.2.6 binbuf\_set()**

```
void binbuf_set (
    t_binbuf * x,
    t_symbol * s,
    short argc,
    t_atom * argv )
```

Use [binbuf\\_set\(\)](#) to change the entire contents of a Binbuf.

The previous contents of the Binbuf are destroyed.

**Parameters**

<i>x</i>	Binbuf to receive the items.
<i>s</i>	Ignored. Pass NULL.
<i>argc</i>	Count of items in the argv array.
<i>argv</i>	Array of t_atoms to put in the Binbuf.

**38.19.2.7 binbuf\_text()**

```
short binbuf_text (
    t_binbuf * x,
    char ** srcText,
    long n )
```

Use [binbuf\\_text\(\)](#) to convert a text handle to a Binbuf.

[binbuf\\_text\(\)](#) parses the text in the handle *srcText* and converts it into binary format. Use it to evaluate a text file or text line entry into a Binbuf.

**Parameters**

<i>x</i>	Binbuf to contain the converted text. It must have already been created with binbuf_new. Its previous contents are destroyed.
<i>srcText</i>	Handle to the text to be converted. It need not be terminated with a 0.
<i>n</i>	Number of characters in the text.

**Returns**

If binbuf\_text encounters an error during its operation, a non-zero result is returned, otherwise it returns 0.

**Remarks**

Note: Commas, symbols containing a dollar sign followed by a number 1-9, and semicolons are identified by special pseudo-type constants for you when your text is binbuf-ized.

The following constants in the a\_type field of Atoms returned by binbuf\_getAtom identify the special symbols [A\\_SEMI](#), [A\\_COMMA](#), and [A\\_DOLLAR](#).

For a [t\\_atom](#) of the pseudo-type [A\\_DOLLAR](#), the a\_w.w\_long field of the [t\\_atom](#) contains the number after the dollar sign in the original text or symbol.

Using these pseudo-types may be helpful in separating 'sentences' and 'phrases' in the input language you design. For example, the old pop-up umenu object allowed users to have spaces in between words by requiring the menu items be separated by commas. It's reasonably easy, using [binbuf\\_getatom\(\)](#), to find the commas in a Binbuf in order to determine the beginning of a new item when reading the atomized text to be displayed in the menu.

If you want to use a literal comma or semicolon in a symbol, precede it with a backslash (\) character. The backslash character can be included by using two backslashes in a row.

**38.19.2.8 binbuf\_totext()**

```
short binbuf_totext (
    t_binbuf * x,
    char ** dstText,
    t_ptr_size * sizep )
```

Use [binbuf\\_totext\(\)](#) to convert a Binbuf into a text handle.

[binbuf\\_totext\(\)](#) converts a Binbuf into text and places it in a handle. Backslashes are added to protect literal commas and semicolons contained in symbols. The pseudo-types are converted into commas, semicolons, or dollar-sign and number, without backslashes preceding them. binbuf\_text can read the output of binbuf\_totext and make the same Binbuf.

**Parameters**

x	Binbuf with data to convert to text.
dstText	Pre-existing handle where the text will be placed. dstText will be resized to accomodate the text.
sizep	Where <a href="#">binbuf_totext()</a> returns the number of characters in the converted text handle.

**Returns**

If binbuf\_totext runs out of memory during its operation, it returns a non-zero result, otherwise it returns 0.

### 38.19.2.9 binbuf\_vinsert()

```
void binbuf_vinsert (
    void * x,
    char * fmt,
    ...
)
```

Use [binbuf\\_vinsert\(\)](#) to append a Max message to a Binbuf adding a semicolon.

#### Parameters

<i>x</i>	Binbuf containing the desired <a href="#">t_atom</a> .
<i>fmt</i>	A C-string containing one or more letters corresponding to the types of each element of the message. s for <a href="#">t_symbol</a> *, l for long, or f for float.
...	Elements of the message, passed directly to the function as Symbols, longs, or floats.

#### Remarks

[binbuf\\_vinsert\(\)](#) works somewhat like a `printf()` for Binbufs. It allows you to pass a number of arguments of different types and insert them into a Binbuf. The entire message will then be terminated with a semicolon. Only 16 items can be passed to [binbuf\\_vinsert\(\)](#).

The example below shows the implementation of a normal object's save method. The save method requires that you build a message that begins with #N (the new object) , followed by the name of your object (in this case, represented by the [t\\_symbol](#) `myobject`), followed by any arguments your instance creation function requires. In this example, we save the values of two fields `m_val1` and `m_val2` defined as longs.

```
void myobject_save (myObject **x, Binbuf *dstBuf)
{
    binbuf_vinsert (dstBuf, "ssll", gensym("#N"),
                    gensym("myobject"),
                    x->m_val1, x->m_val2);
}
```

Suppose that such an object had written this data into a file. If you opened the file as text, you would see the following:

```
#N myobject 10 20;
#P newobj 218 82 30 myobject;
```

The first line will result in a new `myobject` object to be created; the creation function receives the arguments 10 and 20. The second line contains the text of the object box. The `newobj` message to a patcher creates the object box user interface object and attaches it to the previously created `myobject` object. Normally, the `newex` message is used. This causes the object to be created using the arguments that were typed into the object box.

### 38.19.2.10 readatom()

```
short readatom (
    char * outstr,
    char ** text,
    long * n,
    long e,
    t_atom * ap )
```

Use [readatom\(\)](#) to read a single [t\\_atom](#) from a text buffer.

### Parameters

<i>outstr</i>	C-string of 256 characters that will receive the next text item read from the buffer.
<i>text</i>	Handle to the text buffer to be read.
<i>n</i>	Starts at 0, and is modified by <i>readatom</i> to point to the next item in the text buffer.
<i>e</i>	Number of characters in text.
<i>ap</i>	Where the resulting <a href="#">t_atom</a> read from the text buffer is placed.

### Returns

[readatom\(\)](#) returns non-zero if there is more text to read, and zero if it has reached the end of the text. Note that this return value has the opposite logic from that of [binbuf\\_getatom\(\)](#).

### Remarks

This function provides access to the low-level Max text evaluator used by [binbuf\\_text\(\)](#). It is designed to operate on a handle of characters (text) and called in a loop, as in the example shown below.

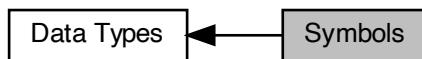
```
long index = 0;
t_atom dst;
char outstr[256];
while (readatom(outstr, textHandle, &index, textLength, &dst))
{
    // do something with the resulting t_atom
}
```

An alternative to using *readatom* is to turn your text into a Binbuf using [binbuf\\_text\(\)](#), then call [binbuf\\_getatom\(\)](#) in a loop.

## 38.20 Symbols

Max maintains a symbol table of all strings to speed lookup for message passing.

Collaboration diagram for Symbols:



### Data Structures

- struct [t\\_symbol](#)

*The symbol.*

## Functions

- `t_symbol * gensym (C74_CONST char *s)`

*Given a C-string, fetch the matching `t_symbol` pointer from the symbol table, generating the symbol if necessary.*

- `t_symbol * gensym_tr (const char *s)`

*Given a C-string, fetch the matching `t_symbol` pointer from the symbol table, generating and translating the symbol if necessary.*

### 38.20.1 Detailed Description

Max maintains a symbol table of all strings to speed lookup for message passing.

If you want to access the bang symbol for example, you'll have to use the expression `gensym("bang")`. For example, `gensym()` may be needed when sending messages directly to other Max objects such as with `object_method()` and `outlet_anything()`. These functions expect a `t_symbol*`, they don't `gensym()` character strings for you.

The `t_symbol` data structure also contains a place to store an arbitrary value. The following example shows how you can use this feature to use symbols to share values among two different external object classes. (Objects of the same class can use the code resource's global variable space to share data.) The idea is that the `s_thing` field of a `t_symbol` can be set to some value, and `gensym()` will return a reference to the Symbol. Thus, the two classes just have to agree about the character string to be used. Alternatively, each could be passed a `t_symbol` that will be used to share data.

Storing a value:

```
t_symbol *s;
s = gensym("some_weird_string");
s->s_thing = (t_object *)someValue;
```

Retrieving a value:

```
t_symbol *s;
s = gensym("some_weird_string");
someValue = s->s_thing;
```

### 38.20.2 Function Documentation

#### 38.20.2.1 gensym()

```
t_symbol* gensym (
    C74_CONST char * s )
```

Given a C-string, fetch the matching `t_symbol` pointer from the symbol table, generating the symbol if necessary.

Parameters

<code>s</code>	A C-string to be looked up in Max's symbol table.
----------------	---

**Returns**

A pointer to the [t\\_symbol](#) in the symbol table.

Referenced by jit\_attr\_offset\_array\_new(), jit\_attr\_offset\_new(), jit\_attribute\_new(), jit\_bin\_read\_matrix(), jit\_bin\_write←\_matrix(), jit\_class\_addinterface(), jit\_class\_method\_addargsafe(), jit\_class\_method\_argsafe\_get(), jit\_mop\_input←\_nolink(), jit\_mop\_io\_new(), jit\_mop\_new(), jit\_mop\_newcopy(), jit\_mop\_output\_nolink(), jit\_mop\_single\_planecount(), jit\_mop\_single\_type(), jit\_object\_exportattrs(), jit\_object\_importattrs(), max\_jit\_attr\_args(), max\_jit\_attr\_get(), max\_jit←\_attr\_getdump(), max\_jit\_mop\_variable\_addinputs(), max\_jit\_mop\_variable\_addoutputs(), max\_jit\_obex\_gimmeback←\_dumpout(), and max\_jit\_obex\_jitob\_set().

**38.20.2.2 gensym\_tr()**

```
t_symbol* gensym_tr (
    const char * s )
```

Given a C-string, fetch the matching [t\\_symbol](#) pointer from the symbol table, generating and translating the symbol if necessary.

**Parameters**

<code>s</code>	A C-string to be looked up in Max's symbol table and then translated
----------------	--

**Returns**

A pointer to the [t\\_symbol](#) in the symbol table.

**38.21 Files and Folders**

These routines assist your object in opening and saving files, as well as locating the user's files in the Max search path.

**Data Structures**

- struct [t\\_fileinfo](#)  
*Information about a file.*
- struct [t\\_path](#)  
*The path data structure.*
- struct [t\\_pathlink](#)  
*The pathlink data structure.*

**Typedefs**

- typedef t\_filestruct \* [t\\_filehandle](#)  
*A t\_filehandle is a cross-platform way of referring to an open file.*

## Enumerations

- enum
 

*The size you should use when allocating strings for full paths.*
- enum
 

*The size you should use when allocating strings for filenames.*
- enum `e_max_path_styles` {
   
`PATH_STYLE_MAX , PATH_STYLE_NATIVE , PATH_STYLE_COLON , PATH_STYLE_SLASH ,`
  
`PATH_STYLE_NATIVE_WIN }`

*Constants that determine the output of `path_nameconform()`.*
- enum `e_max_path_types` {
   
`PATH_TYPE_IGNORE , PATH_TYPE_ABSOLUTE , PATH_TYPE_RELATIVE , PATH_TYPE_BOOT ,`
  
`PATH_TYPE_C74 , PATH_TYPE_PATH , PATH_TYPE_DESKTOP , PATH_TYPE_TILDE ,`
  
`PATH_TYPE_TEMPFOLDER , PATH_TYPE_USERMAX , PATH_TYPE_MAXDB , PATH_TYPE_PLUGIN ,`
  
`PATH_TYPE_PACKAGE }`

*Constants that determine the output of `path_nameconform()`.*
- enum `e_max_fileinfo_flags` { `PATH_FILEINFO_ALIAS , PATH_FILEINFO_FOLDER , PATH_FILEINFO_PACKAGE` }
 

*Flags used to represent properties of a file in a `t_fileinfo` struct.*
- enum `e_max_path_folder_flags` { `PATH_REPORTPACKAGEASFOLDER , PATH_FOLDER_SNIFF , PATH_NOALIASRESOLUTION` }
 

*Flags used by functions such as `path_foldernextfile()` and `path_openfolder()`.*
- enum `e_max_openfile_permissions` { `PATH_READ_PERM , PATH_WRITE_PERM , PATH_RW_PERM` }
 

*Permissions or mode with which to open a file.*
- enum `t_sysfile_pos_mode` { `SYSFILE_ATMARK , SYSFILE_FROMSTART , SYSFILE_FROMEOF ,`
  
`SYSFILE_FROMMARK` }
 

*Modes used by `sysfile_setpos()`*
- enum `t_sysfile_text_flags` {
   
`TEXT_LB_NATIVE , TEXT_LB_MAC , TEXT_LB_PC , TEXT_LB_UNIX ,`
  
`TEXT_LB_MASK , TEXT_ENCODING_USE_FILE , TEXT_NULL_TERMINATE }`

*Flags used reading and writing text files.*

## Functions

- short `path_getapppath` (void)
 

*Retrieve the Path ID of the Max application.*
- short `locatefile` (C74\_CONST char \*name, short \*outvol, short \*binflag)
 

*Find a Max document by name in the search path.*
- short `locatefiletype` (C74\_CONST char \*name, short \*outvol, `t_fourcc` filetype, `t_fourcc` creator)
 

*Find a Max document by name in the search path.*
- short `locatefile_extended` (char \*name, short \*outvol, `t_fourcc` \*outtype, C74\_CONST `t_fourcc` \*filetypelist, short numtypes)
 

*Find a file by name.*
- short `path_resolvefile` (char \*name, C74\_CONST short path, short \*outpath)
 

*Resolve a Path ID plus a (possibly extended) file name into a path that identifies the file's directory and a filename.*
- short `path_fileinfo` (C74\_CONST char \*name, C74\_CONST short path, `t_fileinfo` \*info)
 

*Retrieve a `t_fileinfo` structure from a file/path combination.*
- short `path_topathname` (C74\_CONST short path, C74\_CONST char \*file, char \*name)

- Create a fully qualified file name from a Path ID/file name combination.*
- short `path_frompathname` (C74\_CONST char \*name, short \*path, char \*filename)
 

*Create a filename and Path ID combination from a fully qualified file name.*
  - void `path_setdefault` (short path, short recursive)
 

*Install a path as the default search path.*
  - short `path_getdefault` (void)
 

*Retrieve the Path ID of the default search path.*
  - short `path_getmoddate` (short path, `t_ptr_uint` \*date)
 

*Determine the modification date of the selected path.*
  - short `path_getfilemoddate` (C74\_CONST char \*filename, C74\_CONST short path, `t_ptr_uint` \*date)
 

*Determine the modification date of the selected file.*
  - void \* `path_openfolder` (short path)
 

*Prepare a directory for iteration.*
  - short `path_foldernextfile` (void \*xx, `t_fourcc` \*filetype, char \*name, short descend)
 

*Get the next file in the directory.*
  - void `path_closefolder` (void \*x)
 

*Complete a directory iteration.*
  - short `path_opensysfile` (C74\_CONST char \*name, C74\_CONST short path, `t_filehandle` \*ref, short perm)
 

*Open a file given a filename and Path ID.*
  - short `path_createssysfile` (C74\_CONST char \*name, short path, `t_fourcc` type, `t_filehandle` \*ref)
 

*Create a file given a type code, a filename, and a Path ID.*
  - short `path_nameconform` (C74\_CONST char \*src, char \*dst, long style, long type)
 

*Convert a source path string to destination path string using the specified style and type.*
  - short `path_topotentialname` (C74\_CONST short path, C74\_CONST char \*file, char \*name, short check)
 

*Create a fully qualified file name from a Path ID/file name combination, regardless of whether or not the file exists on disk.*
  - `t_max_err path_toabsolutestempath` (const short in\_path, const char \*in\_filename, char \*out\_filepath)
 

*Translates a Max path+filename combo into a correct POSIX absolute path that can be used to pass to libraries and also handles multiple volumes correctly.*
  - `t_max_err path_absolutepath` (`t_symbol` \*\*returned\_path, const `t_symbol` \*s, const `t_fourcc` \*filetypelist, short numtypes)
 

*Convert a path to an absolutepath as done by the absolutepath object in Max.*
  - long `path_exists` (short path, const char \*filename)
 

*Determine if a path/filename combination exists on disk.*
  - void `open_promptset` (C74\_CONST char \*s)
 

*Use `open_promptset()` to add a prompt message to the open file dialog displayed by `open_dialog()`.*
  - void `saveas_promptset` (C74\_CONST char \*s)
 

*Use `saveas_promptset()` to add a prompt message to the open file dialog displayed by `saveas_dialog()` or `saveasdialog_extended()`.*
  - void \* `filewatcher_new` (`t_object` \*owner, C74\_CONST short path, C74\_CONST char \*filename)
 

*Create a new filewatcher.*
  - void `fileusage_addfile` (void \*w, long flags, C74\_CONST char \*name, short path)
 

*Add a file to a collective.*
  - void `fileusage_addpackage` (void \*w, C74\_CONST char \*name, `t_atomarray` \*subfoldernames)
 

*Add a package to a standalone.*
  - void `fileusage_addfolder` (void \*w, short path, long recursive)
 

*Add a folder to a standalone.*
  - short `open_dialog` (char \*name, short \*volptr, `t_fourcc` \*typeptr, `t_fourcc` \*types, short ntypes)
 

*Present the user with the standard open file dialog.*

- short `saveas_dialog` (char \*filename, short \*path, short \*binptr)  
*Present the user with the standard save file dialog.*
- short `saveasdialog_extended` (char \*name, short \*vol, `t_fourcc` \*type, `t_fourcc` \*typelist, short numtypes)  
*Present the user with the standard save file dialog with your own list of file types.*
- `BEGIN_USING_C_LINKAGE t_max_err sysfile_close (t_filehandle f)`  
*Close a file opened with sysfile\_open().*
- `t_max_err sysfile_read (t_filehandle f, t_ptr_size *count, void *bufptr)`  
*Read a file from disk.*
- `t_max_err sysfile_readtohandle (t_filehandle f, char ***h)`  
*Read the contents of a file into a handle.*
- `t_max_err sysfile_readtoptr (t_filehandle f, char **p)`  
*Read the contents of a file into a pointer.*
- `t_max_err sysfile_write (t_filehandle f, t_ptr_size *count, const void *bufptr)`  
*Write part of a file to disk.*
- `t_max_err sysfile_seteof (t_filehandle f, t_ptr_size logeof)`  
*Set the size of a file handle.*
- `t_max_err sysfile_geteof (t_filehandle f, t_ptr_size *logeof)`  
*Get the size of a file handle.*
- `t_max_err sysfile_setpos (t_filehandle f, t_sysfile_pos_mode mode, t_ptr_int offset)`  
*Set the current file position of a file handle.*
- `t_max_err sysfile_getpos (t_filehandle f, t_ptr_size *filepos)`  
*Get the current file position of a file handle.*
- `t_max_err sysfile_spoolcopy (t_filehandle src, t_filehandle dst, t_ptr_size size)`  
*Copy the contents of one file handle to another file handle.*
- `t_max_err sysfile_readtextfile (t_filehandle f, t_handle htext, t_ptr_size maxlen, t_sysfile_text_flags flags)`  
*Read a text file from disk.*
- `t_max_err sysfile_writetextfile (t_filehandle f, t_handle htext, t_sysfile_text_flags flags)`  
*Write a text file to disk.*
- `t_max_err sysfile_openhandle (char **h, t_sysfile_flags flags, t_filehandle *fh)`  
*Create a `t_filehandle` from a pre-existing handle.*
- `t_max_err sysfile_openptrsize (char *p, t_ptr_size length, t_sysfile_flags flags, t_filehandle *fh)`  
*Create a `t_filehandle` from a pre-existing pointer.*

### 38.21.1 Detailed Description

These routines assist your object in opening and saving files, as well as locating the user's files in the Max search path.

There have been a significant number of changes to these routines (as well as the addition of many functions), so some history may be useful in understanding their use.

Prior to version 4, Max used a feature of Mac OS 9 called "working directories" to specify files. When you used the `locatefile()` service routine, you would get back a file name and a volume number. This name (converted to a Pascal string) and the volume number could be passed to `FSOpen()` to open the located file for reading. The `open_dialog()` routine worked similarly.

In Mac OSX, working directories are no longer supported. In addition, the use of these "volume" numbers makes it somewhat difficult to port Max file routines to other operating systems, such as Windows XP, that specify files using complete pathnames (i.e., "C:\dir1\dir2\file.pat").

However, it is useful to be able to refer to the path and the name of the file separately. The solution involves the retention of the volume number (now called Path ID), but with a platform-independent wrapper that determines its meaning. There are now calls to locate, open, and choose files using C filename strings and Path IDs, as well as routines to convert between a "native" format for specifying a file (such as a full pathname on Windows or an FSRef on the Macintosh) to the C string and Path ID. As of Max version 5 FSSpecs, long ago deprecated by Apple, are no longer supported.

Now that paths in Max have changed to use the slash style, as opposed to the old Macintosh colon style (see the Max 4.3 documentation for a description of the file path styles), there is one function in particular that you will find useful for converting between the various ways paths can be represented, including operating system native paths. This function is [path\\_nameconform\(\)](#). Note that for compatibility purposes Path API functions accept paths in any number of styles, but will typically return paths, or modify paths inline to use the newer slash style. In addition to absolute paths, paths relative to the Max Folder, the "Cycling '74" folder and the boot volume are also supported. See the conformpath.help and ext\_path.h files for more information on the various styles and types of paths. See the "filebyte" SDK example project for a demonstration of how to use the path functions to convert a Max name and path ref pair to a Windows native path for use with CreateFile().

There are a large number of service routine in the Max 4 kernel that support files, but only a handful will be needed by most external objects. In addition to the descriptions that follow, you should consult the movie, folder and filedate examples included with the SDK.

### 38.21.2 The Sysfile API

The Sysfile API provides the means of reading and writing files opened by [path\\_createsysfile\(\)](#) and similar. These functions all make use of an opaque structure, [t\\_filehandle](#). See the path functions [path\\_opensysfile\(\)](#) and [path\\_createsysfile\(\)](#) described earlier in this chapter for more information. The Sysfile API is relatively similar to parts of the old Macintosh File Manager API, and not too different from Standard C library file functions. The "filebyte" example project in the SDK shows how to use these functions to read from a file. It is not safe to mix these routines with other file routines (e.g. don't use fopen() to open a file and [sysfile\\_close\(\)](#) to close it).

In addition to being able to use these routines to write cross-platform code in your max externals, another advantage of the Sysfile API is that it is able to read files stored in the collective file format on both Windows XP and Mac OSX.

### 38.21.3 Example: filebyte (notes from the IRCAM workshop)

#### 38.21.3.1 Paths

- A number that specifies a file location
- Returned by [locatefile\\_extended\(\)](#) and [open\\_dialog\(\)](#)
- Supply a path when opening a file with [path\\_opensysfile\(\)](#)
- Can convert path to and from pathname

#### 38.21.3.2 t\_filehandle

- Returned by [path\\_opensysfile](#)
- Refers to an open file you want to read or write using [sysfile\\_read](#) / [sysfile\\_write](#)
- Could refer to a file in a collective

### 38.21.3.3 File Names

- C string
- Max 5 filenames are UTF-8
- Max 5 supports long (unicode) filenames on both Mac and Windows

### 38.21.3.4 File Path Names

- Max uses a platform-independent path string format: volume:/path1/path2/filename returned by path\_topathname
- Can convert to platform-specific format using path\_nameconform (not needed if using path\_opensysfile)
- Platform-independent format must be used with path\_frompathname

## 38.21.4 Collectives and Fileusage

Use the fileusage routines to add files to a collective when a user chooses to build a collective. Your object can respond to a "fileusage" message, which is sent by Max when the collective builder is building a collective using the following:

```
class_addmethod(c, (method)my_fileusage, "fileusage", A_CANT, 0L);
```

Where my file usage has the prototype:

```
void my_fileusage(t_myObject *x, void *w);
```

Then you can use [fileusage\\_addfile\(\)](#) to add any requisite files to the collective.

## 38.21.5 Filewatchers

Your object can watch a file or folder and be notified of changes. Use [filewatcher\\_new\(\)](#), [filewatcher\\_start\(\)](#), and [filewatcher\\_stop\(\)](#) to implement this functionality. You may wish to use filewatchers sparingly as they can potentially incur computational overhead in the background.

## 38.21.6 Typedef Documentation

### 38.21.6.1 t\_filehandle

```
typedef t_filestruct* t_filehandle
```

A t\_filehandle is a cross-platform way of referring to an open file.

It is an opaque structure, meaning you don't have access to the individual elements of the data structure. You can use a t\_filehandle only with the file routines in the Sysfile API. Do not use other platform-specific file functions in conjunction with these functions. The perm parameter can be either READ\_PERM, WRITE\_PERM, or RW\_PERM.

### 38.21.7 Enumeration Type Documentation

#### 38.21.7.1 anonymous enum

```
anonymous enum
```

The size you should use when allocating strings for filenames.

At the time of this writing it supports up to 256 UTF chars

#### 38.21.7.2 e\_max\_fileinfo\_flags

```
enum e_max_fileinfo_flags
```

Flags used to represent properties of a file in a [t\\_fileinfo](#) struct.

Enumerator

PATH_FILEINFO_ALIAS	alias
PATH_FILEINFO_FOLDER	folder
PATH_FILEINFO_PACKAGE	package (Mac-only)

#### 38.21.7.3 e\_max\_openfile\_permissions

```
enum e_max_openfile_permissions
```

Permissions or mode with which to open a file.

Enumerator

PATH_READ_PERM	Read mode.
PATH_WRITE_PERM	Write mode.
PATH_RW_PERM	Read/Write mode.

#### 38.21.7.4 e\_max\_path\_folder\_flags

```
enum e_max_path_folder_flags
```

Flags used by functions such as [path\\_foldernextfile\(\)](#) and [path\\_openfolder\(\)](#).

## Enumerator

PATH_REPORTPACKAGEASFOLDER	if not true, then a Mac OS package will be reported as a file rather than a folder.
PATH_FOLDER_SNIFF	sniff
PATH_NOALIASRESOLUTION	no alias resolution

**38.21.7.5 e\_max\_path\_styles**

enum [e\\_max\\_path\\_styles](#)

Constants that determine the output of [path\\_nameconform\(\)](#).

## See also

[e\\_max\\_path\\_types](#)

[path\\_nameconform\(\)](#)

## Enumerator

PATH_STYLE_MAX	use PATH_STYLE_MAX_PLAT
PATH_STYLE_NATIVE	use PATH_STYLE_NATIVE_PLAT
PATH_STYLE_COLON	':' sep, "vol:" volume, ":" relative, "^:" boot
PATH_STYLE_SLASH	'/' sep, "vol://" volume, "./" relative, "/" boot
PATH_STYLE_NATIVE_WIN	'\ sep, "vol:\\" volume, ".\\" relative, "\\" boot

**38.21.7.6 e\_max\_path\_types**

enum [e\\_max\\_path\\_types](#)

Constants that determine the output of [path\\_nameconform\(\)](#).

## See also

[e\\_max\\_path\\_styles](#)

[path\\_nameconform\(\)](#)

## Enumerator

PATH_TYPE_IGNORE	ignore
------------------	--------

## Enumerator

PATH_TYPE_ABSOLUTE	absolute path
PATH_TYPE_RELATIVE	relative path
PATH_TYPE_BOOT	boot path
PATH_TYPE_C74	Cycling '74 folder.
PATH_TYPE_PATH	path
PATH_TYPE_DESKTOP	desktop
PATH_TYPE_TILDE	"home"
PATH_TYPE_TEMPFOLDER	/tmp
PATH_TYPE_USERMAX	~/Documents/Max 8
PATH_TYPE_MAXDB	combi: try PATH_TYPE_C74, PATH_TYPE_TILDE, PATH_TYPE_RELATIVE, PATH_TYPE_ABSOLUTE in that order
PATH_TYPE_PLUGIN	AudioUnit/VST plugin (an .auinfo or .vstinfo file in AppSupport)

## 38.21.7.7 t\_sysfile\_pos\_mode

```
enum t_sysfile_pos_mode
```

Modes used by [sysfile\\_setpos\(\)](#)

## Enumerator

SYSFILE_ATMARK	?
SYSFILE_FROMSTART	Calculate the file position from the start of the file.
SYSFILE_FROMLEOF	Calculate the file position from the logical end of the file.
SYSFILE_FROMMARK	Calculate the file position from the current file position.

## 38.21.7.8 t\_sysfile\_text\_flags

```
enum t_sysfile_text_flags
```

Flags used reading and writing text files.

## Enumerator

TEXT_LB_NATIVE	Use the linebreak format native to the current platform.
TEXT_LB_MAC	Use Macintosh line breaks.
TEXT_LB_PC	Use Windows line breaks.
TEXT_LB_UNIX	Use Unix line breaks.
TEXT_ENCODING_USE_FILE	If this flag is not set then the encoding is forced to UTF8.
TEXT_NULL_TERMINATE	Terminate memory returned from <a href="#">sysfile_readtextfile()</a> with a NULL character.

## 38.21.8 Function Documentation

### 38.21.8.1 fileusage\_addfile()

```
void fileusage_addfile (
    void * w,
    long flags,
    C74_CONST char * name,
    short path )
```

Add a file to a collective.

#### Parameters

<i>w</i>	Handle for the collective builder.
<i>flags</i>	If <i>flags</i> == 1, copy this file to support folder of an app instead of to the collective in an app.
<i>name</i>	The name of the file.
<i>path</i>	The path of the file to add.

### 38.21.8.2 fileusage\_addfolder()

```
void fileusage_addfolder (
    void * w,
    short path,
    long recursive )
```

Add a folder to a standalone.

#### Parameters

<i>w</i>	Handle for the standalone builder
<i>path</i>	Path of the folder
<i>recursive</i>	Add the contents of the folder recursively (respected only when building standalones)

#### Version

Introduced in Max 8.0.2

### 38.21.8.3 fileusage\_addpackage()

```
void fileusage_addpackage (
    void * w,
    C74_CONST char * name,
    t_atomarray * subfoldernames )
```

Add a package to a standalone.

#### Parameters

<i>w</i>	Handle for the standalone builder
<i>name</i>	The name of the package
<i>subfoldernames</i>	A <a href="#">t_atomarray</a> containing symbols, each of which is a foldername in the package to include. Pass NULL to include the entire package contents.

#### Version

Introduced in Max 7.0.4

### 38.21.8.4 filewatcher\_new()

```
void* filewatcher_new (
    t_object * owner,
    C74_CONST short path,
    C74_CONST char * filename )
```

Create a new filewatcher.

The file will not be actively watched until `filewatcher_start()` is called. The filewatcher can be freed using [object\\_free\(\)](#).

#### Parameters

<i>owner</i>	Your object. This object will receive the message "filechanged" when the watcher sees a change in the file or folder.
<i>path</i>	The path in which the file being watched resides, or the path of the folder being watched.
<i>filename</i>	The name of the file being watched, or an empty string if you are simply watching the folder specified by path.

#### Returns

A pointer to the new filewatcher.

## Remarks

The "filechanged" method should have the prototype:

```
void myObject_filechanged(t_myObject *x, char *filename, short path);
```

### 38.21.8.5 locatefile()

```
short locatefile (
    C74_CONST char * name,
    short * outvol,
    short * binflag )
```

Find a Max document by name in the search path.

This routine performs the same function as the routine [path\\_getdefault\(\)](#). [locatefile\(\)](#) searches through the directories specified by the user for Patcher files and tables in the File Preferences dialog as well as the current default path (see [path\\_getdefault](#)) and the directory containing the Max application

#### Parameters

<i>name</i>	A C string that is the name of the file to look for.
<i>outvol</i>	The Path ID containing the location of the file if it is found.
<i>binflag</i>	If the file found is in binary format (it's of type 'maxb') 1 is returned here; if it's in text format, 0 is returned.

#### Returns

If a file is found with the name specified by filename, locatefile returns 0, otherwise it returns non-zero.

## Remarks

filename and vol can then be passed to binbuf\_read to read and open file the file. When using MAXplay, the search path consists of all subdirectories of the directory containing the MAXplay application. locatefile only searches for files of type 'maxb' and 'TEXT.'

## See also

[locatefile\\_extended\(\)](#)

### 38.21.8.6 locatefile\_extended()

```
short locatefile_extended (
    char * name,
    short * outvol,
    t_fourcc * outtype,
    C74_CONST t_fourcc * filetypelist,
    short numtypes )
```

Find a file by name.

If a complete path is not specified, search for the name in the search path. This is the preferred method for file searching since its introduction in Max version 4.

#### Version

4.0

#### Parameters

<i>name</i>	The file name for the search, receives actual filename upon return.
<i>outvol</i>	The Path ID of the file (if found).
<i>outtype</i>	The file type of the file (if found).
<i>filetypelist</i>	The file type(s) for which you are searching for, or NULL if any type is acceptable.
<i>numtypes</i>	The number of file types in the typelist array (1 if a single entry, 0 if any type is acceptable).

#### Returns

If a file is found with the name specified by filename, locatefile returns 0, otherwise it returns non-zero.

#### See also

[path\\_getdefault\(\)](#).

#### Remarks

The old file search routines [locatefile\(\)](#) and [locatefiletype\(\)](#) are still supported in Max 4, but the use of a new routine [locatefile\\_extended\(\)](#) is highly recommended. However, [locatefile\\_extended\(\)](#) has an important difference from [locatefile\(\)](#) and [locatefiletype\(\)](#) that may require some rewriting of your code. *It modifies its name parameter in certain cases, while locatefile() and locatefiletype() do not.* The two cases where it could modify the incoming filename string are 1) when an alias is specified, the file pointed to by the alias is returned; and 2) when a full path is specified, the output is the filename plus the path number of the folder it's in.

This is important because many people pass the *s\_name* field of a [t\\_symbol](#) to [locatefile\(\)](#). If the name field of a [t\\_symbol](#) were to be modified, the symbol table would be corrupted. To avoid this problem, use [strncpy\\_zero\(\)](#) to copy the contents of a [t\\_symbol](#) to a character string first, as shown below:

```
char filename[MAX_FILENAME_CHARS];
strncpy_zero(filename,str->s_name, MAX_FILENAME_CHARS);
result = locatefile_extended(filename,&path,&type,filetypelist,1);
```

### 38.21.8.7 locatefiletype()

```
short locatefiletype (
    C74_CONST char * name,
    short * outvol,
    t_fourcc filetype,
    t_fourcc creator )
```

Find a Max document by name in the search path.

This function searches through the same directories as locatefile, but allows you to specify a type and creator of your own.

#### Parameters

<i>name</i>	A C string that is the name of the file to look for.
<i>outvol</i>	The Path ID containing the location of the file if it is found.
<i>filetype</i>	The filetype of the file to look for. If you pass 0L, files of all filetypes are considered.
<i>creator</i>	The creator of the file to look for. If you pass 0L, files with any creator are considered.

#### Returns

If a file is found with the name specified by filename, locatefile returns 0, otherwise it returns non-zero.

#### See also

[locatefile\\_extended\(\)](#)

### 38.21.8.8 open\_dialog()

```
short open_dialog (
    char * name,
    short * volptr,
    t_fourcc * typeptr,
    t_fourcc * types,
    short ntypes )
```

Present the user with the standard open file dialog.

This function is convenient wrapper for using Mac OS Navigation Services or Standard File for opening files.

The mapping of extensions to types is configured in the C74:/init/max-fileformats.txt file. The standard types to use for Max files are 'maxb' for old-format binary files, 'TEXT' for text files, and 'JSON' for newer format patchers or other .json files.

**Parameters**

<i>name</i>	A C-string that will receive the name of the file the user wants to open. The C-string should be allocated with a size of at least #MAX_FILENAME_CHARS.
<i>volptr</i>	Receives the Path ID of the file the user wants to open.
<i>typeptr</i>	The file type of the file the user wants to open.
<i>types</i>	A list of file types to display. This is not limited to 4 types as in the SFGetFile() trap. Pass NULL to display all types.
<i>ntypes</i>	The number of file types in typelist. Pass 0 to display all types.

**Returns**

0 if the user clicked Open in the dialog box. If the user cancelled, [open\\_dialog\(\)](#) returns a non-zero value.

**See also**

[saveasdialog\\_extended\(\)](#)  
[locatefile\\_extended\(\)](#)

**38.21.8.9 open\_promptset()**

```
void open_promptset (
    C74_CONST char * s )
```

Use [open\\_promptset\(\)](#) to add a prompt message to the open file dialog displayed by [open\\_dialog\(\)](#).

Calling this function before [open\\_dialog\(\)](#) permits a string to be displayed in the dialog box instructing the user as to the purpose of the file being opened. It will only apply to the call of [open\\_dialog\(\)](#) that immediately follows [open\\_promptset\(\)](#).

**Parameters**

<i>s</i>	A C-string containing the prompt you wish to display in the dialog box.
----------	---

**Returns**

Ignore.

**See also**

[open\\_dialog\(\)](#)

### 38.21.8.10 path\_absolutepath()

```
t_max_err path_absolutepath (
    t_symbol ** returned_path,
    const t_symbol * s,
    const t_fourcc * filetypeplist,
    short numtypes )
```

Convert a path to an absolutepath as done by the absolutepath object in Max.

#### Parameters

<i>returned_path</i>	Address to a symbol pointer which will be filled-in upon successful return.
<i>s</i>	Incoming (non-absolute) path.
<i>filetypelist</i>	The first of an array of file types to limit the search.
<i>numtypes</i>	The number of entries in the filetypeplist parameter.

#### Returns

A Max error code.

#### See also

[path\\_topathname\(\)](#)  
[locatefile\\_extended\(\)](#)

### 38.21.8.11 path\_closefolder()

```
void path_closefolder (
    void * x )
```

Complete a directory iteration.

#### Parameters

<i>x</i>	The "folder state" value originally returned by <a href="#">path_openfolder()</a> .
----------	---

### 38.21.8.12 path\_createsysfile()

```
short path_createsysfile (
    C74_CONST char * name,
```

```
short path,  
t_fourcc type,  
t_filehandle * ref )
```

Create a file given a type code, a filename, and a Path ID.

#### Parameters

<i>name</i>	The name of the file to be opened.
<i>path</i>	The Path ID of the file to be opened.
<i>type</i>	The file type of the created file.
<i>ref</i>	A <a href="#">t_filehandle</a> reference to the opened file will be returned in this parameter.

#### Returns

An error code.

### 38.21.8.13 path\_exists()

```
long path_exists (  
    short path,  
    const char * filename )
```

Determine if a path/filename combination exists on disk.

#### Parameters

<i>path</i>	The Max path reference.
<i>filename</i>	The name of the file in that path to test.

#### Returns

1 = the specified file exists, 0 = the specified file does not exist.

#### See also

[path\\_topathname\(\)](#)  
[path\\_topotentialname\(\)](#)

### 38.21.8.14 path\_fileinfo()

```
short path_fileinfo (
    C74_CONST char * name,
    C74_CONST short path,
    t_fileinfo * info )
```

Retrive a [t\\_fileinfo](#) structure from a file/path combination.

#### Parameters

<i>name</i>	The file name to be queried.
<i>path</i>	The Path ID of the file.
<i>info</i>	The address of a <a href="#">t_fileinfo</a> structure to contain the file information.

#### Returns

Returns 0 if successful, otherwise it returns an OS-specific error code.

### 38.21.8.15 path\_foldernextfile()

```
short path_foldernextfile (
    void * xx,
    t_fourcc * filetype,
    char * name,
    short descend )
```

Get the next file in the directory.

In conjunction with [path\\_openfolder\(\)](#) and [path\\_closefolder\(\)](#), this routine allows you to iterate through all of the files in a path.

#### Parameters

<i>xx</i>	The "folder state" value returned by <a href="#">path_openfolder()</a> .
<i>filetype</i>	Contains the file type of the file type on return.
<i>name</i>	Contains the file name of the next file on return.
<i>descend</i>	Unused.

#### Returns

Returns non-zero if successful, and zero when there are no more files.

**See also**[e\\_max\\_path\\_folder\\_flags](#)**38.21.8.16 path\_frompathname()**

```
short path_frompathname (
    C74_CONST char * name,
    short * path,
    char * filename )
```

Create a filename and Path ID combination from a fully qualified file name.

Note that [path\\_frompathname\(\)](#) does not require that the file actually exist. In this way you can use it to convert a full path you may have received as an argument to a file writing message to a form appropriate to provide to a routine such as [path\\_createfile\(\)](#).

**Parameters**

<i>name</i>	The extended file path to be converted.
<i>path</i>	Contains the Path ID on return.
<i>filename</i>	Contains the file name on return.

**Returns**

Returns 0 if successful, otherwise it returns an OS-specific error code.

**38.21.8.17 path\_getapppath()**

```
short path_getapppath (
    void )
```

Retrieve the Path ID of the Max application.

**Returns**

The path id.

### 38.21.8.18 path\_getdefault()

```
short path_getdefault (
    void )
```

Retrieve the Path ID of the default search path.

#### Returns

The path id of the default search path.

### 38.21.8.19 path\_getfilemoddate()

```
short path_getfilemoddate (
    C74_CONST char * filename,
    C74_CONST short path,
    t_ptr_uint * date )
```

Determine the modification date of the selected file.

#### Parameters

<i>filename</i>	The name of the file to query.
<i>path</i>	The Path ID of the file.
<i>date</i>	The last modification date of the file upon return.

#### Returns

An error code.

### 38.21.8.20 path\_getmoddate()

```
short path_getmoddate (
    short path,
    t_ptr_uint * date )
```

Determine the modification date of the selected path.

#### Parameters

<i>path</i>	The Path ID of the directory to check.
<i>date</i>	The last modification date of the directory.

**Returns**

An error code.

**38.21.8.21 path\_nameconform()**

```
short path_nameconform (
    C74_CONST char * src,
    char * dst,
    long style,
    long type )
```

Convert a source path string to destination path string using the specified style and type.

**Parameters**

<i>src</i>	A pointer to source character string to be converted.
<i>dst</i>	A pointer to destination character string.
<i>style</i>	The destination filepath style, as defined in <a href="#">e_max_path_styles</a>
<i>type</i>	The destination filepath type, as defined in <a href="#">e_max_path_types</a>

**Returns**

An error code.

**See also**

[#MAX\\_PATH\\_CHARS](#)

**38.21.8.22 path\_openfolder()**

```
void* path_openfolder (
    short path )
```

Prepare a directory for iteration.

**Parameters**

<i>path</i>	The directory Path ID to open.
-------------	--------------------------------

**Returns**

The return value of this routine is an internal "folder state" structure used for further folder manipulation. It should be saved and used for calls to [path\\_foldernextfile\(\)](#) and [path\\_closefolder\(\)](#). If the folder cannot be found or accessed, [path\\_openfolder\(\)](#) returns 0.

**38.21.8.23 path\_opensysfile()**

```
short path_opensysfile (
    C74_CONST char * name,
    C74_CONST short path,
    t_filehandle * ref,
    short perm )
```

Open a file given a filename and Path ID.

**Parameters**

<i>name</i>	The name of the file to be opened.
<i>path</i>	The Path ID of the file to be opened.
<i>ref</i>	A <a href="#">t_filehandle</a> reference to the opened file will be returned in this parameter.
<i>perm</i>	The permission for the opened file as defined in <a href="#">e_max_openfile_permissions</a> .

**Returns**

An error code.

**38.21.8.24 path\_resolvefile()**

```
short path_resolvefile (
    char * name,
    C74_CONST short path,
    short * outpath )
```

Resolve a Path ID plus a (possibly extended) file name into a path that identifies the file's directory and a filename.

This routine converts a name and Path ID to a standard form in which the name has no path information and does not refer to an aliased file.

**Parameters**

<i>name</i>	A file name (which may be fully or partially qualified), will contain the file name on return.
<i>path</i>	The Path ID to be resolved.
<i>outpath</i>	The Path ID of the returned file name.

**Returns**

Returns 0 if successful.

**38.21.8.25 path\_setdefault()**

```
void path_setdefault (
    short path,
    short recursive )
```

Install a path as the default search path.

The default path is searched before the Max search path. For instance, when loading a patcher from a directory outside the search path, the patcher's directory is searched for files before the search path. [path\\_setdefault\(\)](#) allows you to set a path as the default.

**Parameters**

<i>path</i>	The path to use as the search path. If path is already part of the Max Search path, it will not be added (since, by default, it will be searched during file searches).
<i>recursive</i>	If non-zero, all subdirectories will be installed in the default search list. Be very careful with the use of the recursive argument. It has the capacity to slow down file searches dramatically as the list of folders is being built. Max itself never creates a hierarchical default search path.

**38.21.8.26 path\_toabsolutesystempath()**

```
t_max_err path_toabsolutesystempath (
    const short in_path,
    const char * in_filename,
    char * out_filepath )
```

Translates a Max path+filename combo into a correct POSIX absolute path that can be used to pass to libraries and also handles multiple volumes correctly.

**Parameters**

<i>in_path</i>	The Max path reference
<i>in_filename</i>	The name of the file in that path.
<i>out_filepath</i>	A string that is MAX_PATH_CHARS in length, which will receive the formatted absolute path upon return.

**Returns**

Returns 0 if successful.

**See also**

[path\\_topotentialname\(\)](#)  
[path\\_nameconform\(\)](#)

```
// example for getting a windows-formatted path for a folder path:
path_toabsoletesystempath(pathid, "", filestring);
path_nameconform(filestring, sNativeQualifiedPathname, PATH_STYLE_NATIVE, PATH_TYPE_PATH);
```

**38.21.8.27 path\_topathname()**

```
short path_topathname (
    C74_CONST short path,
    C74_CONST char * file,
    char * name )
```

Create a fully qualified file name from a Path ID/file name combination.

Unlike [path\\_topotentialname\(\)](#), this routine will only convert a pathname pair to a valid path string if the path exists.

**Parameters**

<i>path</i>	The path to be used.
<i>file</i>	The file name to be used.
<i>name</i>	Loaded with the fully extended file name on return.

**Returns**

Returns 0 if successful, otherwise it returns an OS-specific error code.

**38.21.8.28 path\_topotentialname()**

```
short path_topotentialname (
    C74_CONST short path,
    C74_CONST char * file,
    char * name,
    short check )
```

Create a fully qualified file name from a Path ID/file name combination, regardless of whether or not the file exists on disk.

**Parameters**

<i>path</i>	The path to be used.
<i>file</i>	The file name to be used.
<i>name</i>	Loaded with the fully extended file name on return.
<i>check</i>	Flag to check if a file with the given path exists.

**Returns**

Returns 0 if successful, otherwise it returns an OS-specific error code.

**See also**

[path\\_topathname\(\)](#)

**38.21.8.29 saveas\_dialog()**

```
short saveas_dialog (
    char * filename,
    short * path,
    short * binptr )
```

Present the user with the standard save file dialog.

The mapping of extensions to types is configured in the C74:/init/max-fileformats.txt file. The standard types to use for Max files are 'maxb' for old-format binary files, 'TEXT' for text files, and 'JSON' for newer format patchers or other .json files.

**Parameters**

<i>filename</i>	A C-string containing a default name for the file to save. If the user decides to save a file, its name is returned here. The C-string should be allocated with a size of at least #MAX_FILENAME_CHARS.
<i>path</i>	If the user decides to save the file, the Path ID of the location chosen is returned here.
<i>binptr</i>	Pass NULL for this parameter. This parameter was used in Max 4 to allow the choice of saving binary or text format patchers.

**Returns**

0 if the user choose to save the file. If the user cancelled, returns a non-zero value.

**See also**

[open\\_dialog\(\)](#)  
[saveasdialog\\_extended\(\)](#)  
[locatefile\\_extended\(\)](#)

**38.21.8.30 saveas\_promptset()**

```
void saveas_promptset (
    C74_CONST char * s )
```

Use [saveas\\_promptset\(\)](#) to add a prompt message to the open file dialog displayed by [saveas\\_dialog\(\)](#) or [saveasdialog\\_extended\(\)](#).

Calling this function before [saveasdialog\\_extended\(\)](#) permits a string to be displayed in the dialog box instructing the user as to the purpose of the file being opened. It will only apply to the call of [saveasdialog\\_extended\(\)](#) that immediately follows [saveas\\_promptset\(\)](#).

**Parameters**

<i>s</i>	A C-string containing the prompt you wish to display in the dialog box.
----------	---

**Returns**

Ignore.

**See also**

[open\\_dialog\(\)](#)

**38.21.8.31 saveasdialog\_extended()**

```
short saveasdialog_extended (
    char * name,
    short * vol,
    t_fourcc * type,
    t_fourcc * typelist,
    short numtypes )
```

Present the user with the standard save file dialog with your own list of file types.

[saveasdialog\\_extended\(\)](#) is similar to [saveas\\_dialog\(\)](#), but allows the additional feature of specifying a list of possible types. These will be displayed in a pop-up menu.

File types found in the typelist argument that match known Max types will be displayed with descriptive text. Unmatched types will simply display the type name (for example, "foXx" is not a standard type so it would be shown in the pop-up menu as foXx)

Known file types include:

- TEXT: text file
- maxb: Max binary patcher
- maxc: Max collective
- Midi: MIDI file
- Sd2f: Sound Designer II audio file
- NxTS: NeXT/Sun audio file
- WAVE: WAVE audio file.
- AIFF: AIFF audio file
- mP3f: Max preference file
- PICT: PICT graphic file
- MooV: Quicktime movie file
- aPcs: VST plug-in
- AFxP: VST effect patch data file
- AFxB: VST effect bank data file
- DATA: Raw data audio file
- ULAW: NeXT/Sun audio file

#### Parameters

<i>name</i>	A C-string containing a default name for the file to save. If the user decides to save a file, its name is returned here. The C-string should be allocated with a size of at least #MAX_FILENAME_CHARS.
<i>vol</i>	If the user decides to save the file, the Path ID of the location chosen is returned here.
<i>type</i>	Returns the type of file chosen by the user.
<i>typelist</i>	The list of types provided to the user.
<i>numtypes</i>	The number of file types in typelist.

#### Returns

0 if the user choose to save the file. If the user cancelled, returns a non-zero value.

#### See also

[open\\_dialog\(\)](#)  
[locatefile\\_extended\(\)](#)

### 38.21.8.32 sysfile\_close()

```
BEGIN_USING_C_LINKAGE t_max_err sysfile_close (
    t_filehandle f )
```

Close a file opened with sysfile\_open().

This function is similar to FSClose() or fclose(). It should be used instead of system-specific file closing routines in order to make max external code that will compile cross-platform.

#### Parameters

<i>f</i>	The <a href="#">t_filehandle</a> structure of the file the user wants to close.
----------	---

#### Returns

An error code.

### 38.21.8.33 sysfile\_geteof()

```
t_max_err sysfile_geteof (
    t_filehandle f,
    t_ptr_size * logeof )
```

Get the size of a file handle.

This function is similar to and should be used instead of GetEOF(). The function gets the size of file handle in bytes, and places it in *logeof*.

#### Parameters

<i>f</i>	The file's <a href="#">t_filehandle</a> structure.
<i>logeof</i>	The file size in bytes is returned to this value.

#### Returns

An error code.

### 38.21.8.34 sysfile\_getpos()

```
t_max_err sysfile_getpos (
    t_filehandle f,
    t_ptr_size * filepos )
```

Get the current file position of a file handle.

This function is similar to and should be used instead of GetFPos(). The function gets the current file position of file handle in bytes, and places it in "filepos".

#### Parameters

<i>f</i>	The file's <a href="#">t_filehandle</a> structure.
<i>filepos</i>	The address of a variable to hold the current file position of file handle in bytes.

#### Returns

An error code.

### 38.21.8.35 sysfile\_openhandle()

```
t_max_err sysfile_openhandle (
    char ** h,
    t_sysfile_flags flags,
    t_filehandle * fh )
```

Create a [t\\_filehandle](#) from a pre-existing handle.

#### Parameters

<i>h</i>	A handle for some data, data is <i>*not*</i> copied and <i>*not*</i> freed on file close.
<i>flags</i>	Pass 0 (additional flags are private).
<i>fh</i>	The address of a <a href="#">t_filehandle</a> which will be allocated.

#### Returns

An error code.

### 38.21.8.36 sysfile\_openptrsize()

```
t_max_err sysfile_openptrsize (
    char * p,
    t_ptr_size length,
    t_sysfile_flags flags,
    t_filehandle * fh )
```

Create a [t\\_filehandle](#) from a pre-existing pointer.

**Parameters**

<i>p</i>	A pointer to some data. Data is <i>*not*</i> copied and <i>*not*</i> freed on file close.
<i>length</i>	The size of <i>p</i> .
<i>flags</i>	Pass 0 (additional flags are private).
<i>fh</i>	The address of a <a href="#">t_filehandle</a> which will be allocated.

**Returns**

An error code.

**38.21.8.37 sysfile\_read()**

```
t_max_err sysfile_read (
    t_filehandle f,
    t_ptr_size * count,
    void * bufptr )
```

Read a file from disk.

This function is similar to FSRead() or fread(). It should be used instead of these functions (or other system-specific file reading routines) in order to make max external code that will compile cross-platform. It reads "count" bytes from file handle at current file position into "bufptr". The byte count actually read is set in "count", and the file position is updated by the actual byte count read.

**Parameters**

<i>f</i>	The <a href="#">t_filehandle</a> structure of the file the user wants to open.
<i>count</i>	Pointer to the number of bytes that will be read from the file at the current file position. The byte count actually read is returned to this value.
<i>bufptr</i>	Pointer to the buffer that the data will be read into.

**Returns**

An error code.

Referenced by jit\_bin\_read\_chunk\_info(), jit\_bin\_read\_header(), and jit\_bin\_read\_matrix().

**38.21.8.38 sysfile\_readtextfile()**

```
t_max_err sysfile_readtextfile (
    t_filehandle f,
```

```
t_handle htext,
t_ptr_size maxlen,
t_sysfile_text_flags flags )
```

Read a text file from disk.

This function reads up to the maximum number of bytes given by maxlen from file handle at current file position into the htext handle, performing linebreak translation if set in flags.

#### Parameters

<i>f</i>	The <a href="#">t_filehandle</a> structure of the text file the user wants to open.
<i>htext</i>	Handle that the data will be read into.
<i>maxlen</i>	The maximum length in bytes to be read into the handle. Passing the value 0L indicates no maximum (i.e. read the entire file).
<i>flags</i>	Flags to set linebreak translation as defined in <a href="#">t_sysfile_text_flags</a> .

#### Returns

An error code.

### 38.21.8.39 sysfile\_readtohandle()

```
t_max_err sysfile_readtohandle (
    t_filehandle f,
    char *** h )
```

Read the contents of a file into a handle.

#### Parameters

<i>f</i>	The open <a href="#">t_filehandle</a> structure to read into the handle.
<i>h</i>	The address of a handle into which the file will be read.

#### Returns

An error code.

#### Remarks

You should free the pointer, when you are done with it, using [sysmem\\_freehandle\(\)](#).

### 38.21.8.40 sysfile\_readtoptr()

```
t_max_err sysfile_readtoptr (
    t_filehandle f,
    char ** p )
```

Read the contents of a file into a pointer.

#### Parameters

<i>f</i>	The open <a href="#">t_filehandle</a> structure to read into the handle.
<i>p</i>	The address of a pointer into which the file will be read.

#### Returns

An error code.

#### Remarks

You should free the pointer, when you are done with it, using [sysmem\\_freeptr\(\)](#).

### 38.21.8.41 sysfile\_seteof()

```
t_max_err sysfile_seteof (
    t_filehandle f,
    t_ptr_size logeof )
```

Set the size of a file handle.

This function is similar to and should be used instead of SetEOF(). The function sets the size of file handle in bytes, specified by ♦“logeof”?

#### Parameters

<i>f</i>	The file's <a href="#">t_filehandle</a> structure.
<i>logeof</i>	The file size in bytes.

#### Returns

An error code.

### 38.21.8.42 sysfile\_setpos()

```
t_max_err sysfile_setpos (
    t_filehandle f,
    t_sysfile_pos_mode mode,
    t_ptr_int offset )
```

Set the current file position of a file handle.

This function is similar to and should be used instead of SetFPos(). It is used to set the current file position of file handle to by the given number of offset bytes relative to the mode used, as defined in [t\\_sysfile\\_pos\\_mode](#).

#### Parameters

<i>f</i>	The file's <a href="#">t_filehandle</a> structure.
<i>mode</i>	Mode from which the offset will be calculated, as defined in <a href="#">t_sysfile_pos_mode</a> .
<i>offset</i>	The offset in bytes relative to the mode.

#### Returns

An error code.

Referenced by [jit\\_bin\\_read\\_chunk\\_info\(\)](#), [jit\\_bin\\_read\\_header\(\)](#), and [jit\\_bin\\_write\\_header\(\)](#).

### 38.21.8.43 sysfile\_spoolcopy()

```
t_max_err sysfile_spoolcopy (
    t_filehandle src,
    t_filehandle dst,
    t_ptr_size size )
```

Copy the contents of one file handle to another file handle.

#### Parameters

<i>src</i>	The file handle from which to copy.
<i>dst</i>	The file handle to which the copy is written.
<i>size</i>	The number of bytes to copy. If 0 the size of src will be used.

#### Returns

An error code.

### 38.21.8.44 sysfile\_write()

```
t_max_err sysfile_write (
    t_filehandle f,
    t_ptr_size * count,
    const void * bufptr )
```

Write part of a file to disk.

This function is similar to FSWrite() or fwrite(). It should be used instead of these functions (or other system-specific file reading routines) in order to make max external code that will compile cross-platform. The function writes "count" bytes from "bufptr" into file handle at current file position. The byte count actually written is set in "count", and the file position is updated by the actual byte count written.

#### Parameters

<i>f</i>	The <code>t_filehandle</code> structure of the file to which the user wants to write.
<i>count</i>	Pointer to the number of bytes that will be written to the file at the current file position. The byte count actually written is returned to this value.
<i>bufptr</i>	Pointer to the buffer that the data will be read from.

#### Returns

An error code.

Referenced by `jit_bin_write_header()`, and `jit_bin_write_matrix()`.

### 38.21.8.45 sysfile\_writetextfile()

```
t_max_err sysfile_writetextfile (
    t_filehandle f,
    t_handle htext,
    t_sysfile_text_flags flags )
```

Write a text file to disk.

This function writes a text handle to a text file performing linebreak translation if set in flags.

#### Parameters

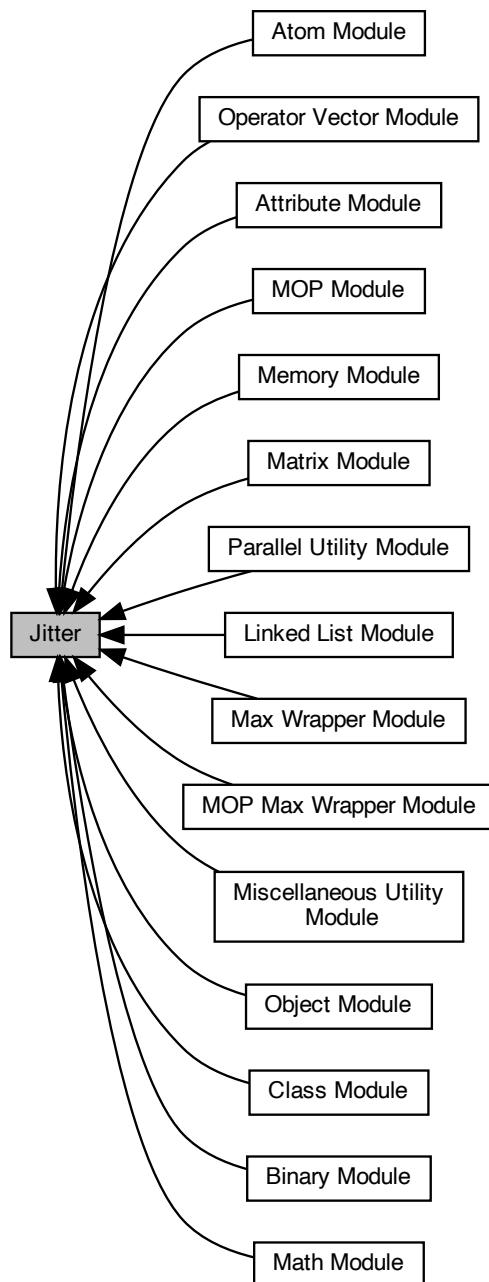
<i>f</i>	The <code>t_filehandle</code> structure of the text file to which the user wants to write.
<i>htext</i>	Handle that the data that will be read from.
<i>flags</i>	Flags to set linebreak translation as defined in <code>t_sysfile_text_flags</code> .

**Returns**

An error code.

## 38.22 Jitter

Collaboration diagram for Jitter:



## Modules

- Atom Module
- Attribute Module
- Binary Module
- Class Module
- Object Module
- Miscellaneous Utility Module
- Linked List Module
- Math Module
- Matrix Module
- Max Wrapper Module
- Memory Module
- MOP Module
- Parallel Utility Module
- MOP Max Wrapper Module
- Operator Vector Module

## Typedefs

- `typedef t_object t_jit_object`  
*object header*

## Variables

- `JIT_EX_DATA t_symbol * _jit_sym_nothing`  
*cached t\_symbol*
- `JIT_EX_DATA t_symbol * _jit_sym_new`  
*cached t\_symbol*
- `JIT_EX_DATA t_symbol * _jit_sym_free`  
*cached t\_symbol*
- `JIT_EX_DATA t_symbol * _jit_sym_classname`  
*cached t\_symbol*
- `JIT_EX_DATA t_symbol * _jit_sym_getname`  
*cached t\_symbol*
- `JIT_EX_DATA t_symbol * _jit_sym_getmethod`  
*cached t\_symbol*
- `JIT_EX_DATA t_symbol * _jit_sym_get`  
*cached t\_symbol*
- `JIT_EX_DATA t_symbol * _jit_sym_set`  
*cached t\_symbol*
- `JIT_EX_DATA t_symbol * _jit_sym_register`  
*cached t\_symbol*
- `JIT_EX_DATA t_symbol * _jit_sym_char`  
*cached t\_symbol*
- `JIT_EX_DATA t_symbol * _jit_sym_long`  
*cached t\_symbol*

- JIT\_EX\_DATA `t_symbol * _jit_sym_float32`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_float64`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_symbol`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_pointer`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_object`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_atom`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_list`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_type`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_dim`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_planecount`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_val`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_plane`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_cell`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_jit_matrix`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_class_jit_matrix`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_togworld`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_fromgworld`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_frommatrix`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_class_jit_attribute`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_jit_attribute`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_jit_attr_offset`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_jit_attr_offset_array`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_rebuilding`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_modified`

- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_lock*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_setinfo*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_setinfo\_ex*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_getinfo*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_data*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_getdata*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_outputmatrix*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_clear*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_clear\_custom*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_err\_calculate*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_max\_jit\_classex*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_setall*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_chuck*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_getsize*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_getindex*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_objptr2index*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_append*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_insertindex*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_deleteindex*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_chuckindex*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_makearray*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_reverse*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_rotate*  
        *cached t\_symbol*

- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_shuffle`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_swap`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_findfirst`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_findall`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_methodall`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_methodindex`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_sort`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_matrix_calc`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_genframe`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_filter`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_jit_mop`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_newcopy`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_jit_linklist`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_inputcount`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_outputcount`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_getinput`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_getoutput`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_getinputlist`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_getoutputlist`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_ioname`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_matrixname`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_outputmode`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_matrix`  
    *cached t\_symbol*
- JIT\_EX\_DATA `t_symbol` \* `_jit_sym_getmatrix`

- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_typealink*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_dimlink*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_planelink*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_restrict\_type*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_restrict\_planecount*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_restrict\_dim*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_special*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_getspecial*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_adapt*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_decorator*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_frommatrix\_trunc*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_ioproc*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_getioproc*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_name*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_types*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_minplaneCount*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_maxplaneCount*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_minDimCount*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_maxDimCount*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_minDim*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_maxDim*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_gl\_points*  
        *cached t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* *\_jit\_sym\_gl\_point\_sprite*  
        *cached t\_symbol*

- JIT\_EX\_DATA `t_symbol * _jit_sym_gl_lines`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_gl_line_strip`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_gl_line_loop`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_gl_triangles`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_gl_tri_strip`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_gl_tri_fan`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_gl_quads`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_gl_quad_strip`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_gl_polygon`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_gl_tri_grid`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_gl_quad_grid`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_err_lockout_stack`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_class_jit_namespace`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_jit_namespace`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_findsize`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_attach`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_detach`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_add`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_replace`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_gettype`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_ob_sym`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_resolve_name`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_resolve_raw`  
*cached t\_symbol*
- JIT\_EX\_DATA `t_symbol * _jit_sym_notifyall`

- JIT\_EX\_DATA *t\_symbol* \* \_jit\_sym\_block
  - cached *t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* \_jit\_sym\_unblock
  - cached *t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* \_jit\_sym\_position
  - cached *t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* \_jit\_sym\_rotatexyz
  - cached *t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* \_jit\_sym\_scale
  - cached *t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* \_jit\_sym\_quat
  - cached *t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* \_jit\_sym\_direction
  - cached *t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* \_jit\_sym\_lookat
  - cached *t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* \_jit\_sym\_anim
  - cached *t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* \_jit\_sym\_bounds
  - cached *t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* \_jit\_sym\_boundcalc
  - cached *t\_symbol*
- JIT\_EX\_DATA *t\_symbol* \* \_jit\_sym\_calcbounds
  - cached *t\_symbol*

### 38.22.1 Detailed Description

## 38.23 Memory Management

In the past, Max has provided two separate APIs for memory management.

### Macros

- #define MM\_UNIFIED

*This macro being defined means that getbytes and sysmem APIs for memory management are unified.*

## Functions

- `char * getbytes (t_getbytes_size size)`  
*Allocate small amounts of non-relocatable memory.*
- `void freebytes (void *b, t_getbytes_size size)`  
*Free memory allocated with `getbytes()`.*
- `char * getbytes16 (t_getbytes_size size)`  
*Use `getbytes16()` to allocate small amounts of non-relocatable memory that is aligned on a 16-byte boundary for use with vector optimization.*
- `void freebytes16 (char *mem, t_getbytes_size size)`  
*Free memory allocated with `getbytes16()`.*
- `char ** newhandle (long size)`  
*Allocate relocatable memory.*
- `short growhandle (void *h, long size)`  
*Change the size of a handle.*
- `void disphoshandle (char **h)`  
*Free the memory used by a handle you no longer need.*
- `BEGIN_USING_C_LINKAGE t_ptr sysmem_newptr (t_ptr_size size)`  
*Allocate memory.*
- `t_ptr sysmem_newptrclear (t_ptr_size size)`  
*Allocate memory and set it to zero.*
- `t_ptr sysmem_resizeptr (void *ptr, t_ptr_size newsize)`  
*Resize an existing pointer.*
- `t_ptr sysmem_resizeptrclear (void *ptr, t_ptr_size newsize)`  
*Resize an existing pointer and clear the newly allocated memory, if any.*
- `t_ptr_size sysmem_ptrsize (void *ptr)`  
*Find the size of a pointer.*
- `void sysmem_freeptr (void *ptr)`  
*Free memory allocated with `sysmem_newptr()`.*
- `void sysmem_copyptr (const void *src, void *dst, t_ptr_size bytes)`  
*Copy memory the contents of one pointer to another pointer.*
- `t_handle sysmem_newhandle (t_ptr_size size)`  
*Allocate a handle (a pointer to a pointer).*
- `t_handle sysmem_newhandleclear (t_ptr_size size)`  
*Allocate a handle (a pointer to a pointer) whose memory is set to zero.*
- `t_max_err sysmem_resizehandle (t_handle handle, t_ptr_size newsize)`  
*Resize an existing handle.*
- `t_ptr_size sysmem_handlesize (t_handle handle)`  
*Find the size of a handle.*
- `void sysmem_freehandle (t_handle handle)`  
*Free memory allocated with `sysmem_newhandle()`.*
- `int sysmem_lockhandle (t_handle handle, int lock)`  
*Set the locked/unlocked state of a handle.*
- `t_max_err sysmem_ptrandhand (void *p, t_handle h, t_ptr_size size)`  
*Add memory to an existing handle and copy memory to the resized portion from a pointer.*
- `t_max_err sysmem_ptrbeforehand (void *p, t_handle h, t_ptr_size size)`  
*Add memory to an existing handle and copy memory to the resized portion from a pointer.*
- `t_max_err sysmem_nullterminatehandle (t_handle h)`  
*Add a null terminator to a handle.*

### 38.23.1 Detailed Description

In the past, Max has provided two separate APIs for memory management.

One for allocating memory on the stack so that it was interrupt safe, including the [getbytes\(\)](#) and [freebytes\(\)](#) functions. The other, the "sysmem" API, were for allocating memory on the heap where larger amounts of memory were needed and the code could be guaranteed to operate at non-interrupt level.

Many things have changed in the environment of recent operating systems (MacOS X and Windows XP/Vista), the memory routines function differently, and the scheduler is no longer directly triggered by a hardware interrupt. In Max 5, the sysmem and getbytes API's have been unified, and thus may be used interchangeably.

The memory management unification can be switched on and off in the header files if needed, to compile code for older versions of Max for example, by changing the use of [MM\\_UNIFIED](#) in the Max headers.

### 38.23.2 Sysmem API

The Sysmem API provides a number of utilities for allocating and managing memory. It is relatively similar to some of the Macintosh Memory Manager API, and not too different from Standard C library memory functions. It is *not* safe to mix these routines with other memory routines (e.g. don't use malloc() to allocate a pointer, and [sysmem\\_freeptr\(\)](#) to free it).

### 38.23.3 Macro Definition Documentation

#### 38.23.3.1 MM\_UNIFIED

```
#define MM_UNIFIED
```

This macro being defined means that getbytes and sysmem APIs for memory management are unified.

This is correct for Max 5, but should be commented out when compiling for older max targets.

### 38.23.4 Function Documentation

#### 38.23.4.1 disposthandle()

```
void disposthandle (
    char ** h )
```

Free the memory used by a handle you no longer need.

Deprecated, use sysmem\_freehandle instead

**Parameters**

<i>h</i>	The handle to dispose.
----------	------------------------

**See also**

[sysmem\\_freehandle\(\)](#)

**38.23.4.2 freebytes()**

```
void freebytes (
    void * b,
    t_getbytes_size size )
```

Free memory allocated with [getbytes\(\)](#).

As of Max 5 it is unified with [sysmem\\_newptr\(\)](#), which is the preferred method for allocating memory.

**Parameters**

<i>b</i>	A pointer to the block of memory previously allocated that you want to free.
<i>size</i>	The size the block specified (as parameter <i>b</i> ) in bytes.

Referenced by `jit_object_exportattrs()`, `jit_object_importattrs()`, `max_jit_attr_getdump()`, `max_jit_obex_gimmeback()`, and `max_jit_obex_gimmeback_dumpout()`.

**38.23.4.3 freebytes16()**

```
void freebytes16 (
    char * mem,
    t_getbytes_size size )
```

Free memory allocated with [getbytes16\(\)](#).

As of Max 5 it is unified with [sysmem\\_newptr\(\)](#), which is the preferred method for allocating memory.

**Parameters**

<i>mem</i>	A pointer to the block of memory previously allocated that you want to free.
<i>size</i>	The size the block specified (as parameter <i>b</i> ) in bytes.

**Remarks**

Note that [freebytes16\(\)](#) will cause memory corruption if you pass it memory that was allocated with [getbytes\(\)](#). Use it only with memory allocated with [getbytes16\(\)](#).

**38.23.4.4 getbytes()**

```
char* getbytes (
    t_getbytes_size size )
```

Allocate small amounts of non-relocatable memory.

As of Max 5 it is unified with [sysmem\\_newptr\(\)](#), which is the preferred method for allocating memory.

**Parameters**

<i>size</i>	The size to allocate in bytes (up to 32767 bytes).
-------------	--

**Returns**

A pointer to the allocated memory.

**38.23.4.5 getbytes16()**

```
char* getbytes16 (
    t_getbytes_size size )
```

Use [getbytes16\(\)](#) to allocate small amounts of non-relocatable memory that is aligned on a 16-byte boundary for use with vector optimization.

**Parameters**

<i>size</i>	The size to allocate in bytes (up to 32767 bytes).
-------------	--

**Returns**

A pointer to the allocated memory.

**Remarks**

[getbytes16\(\)](#) is identical to [getbytes](#) except that it returns memory that is aligned to a 16-byte boundary. This allows you to allocate storage for vector-optimized memory at interrupt level. Note that any memory allocated with [getbytes16\(\)](#) must be freed with [freebytes16\(\)](#), not [freebytes\(\)](#).

**38.23.4.6 growhandle()**

```
short growhandle (
    void * h,
    long size )
```

Change the size of a handle.

Deprecated, use [sysmem\\_resizehandle](#) instead

**Parameters**

<i>h</i>	The handle to resize.
<i>size</i>	The new size to allocate in bytes.

**Returns**

Ignored.

**See also**

[sysmem\\_resizehandle\(\)](#)

**38.23.4.7 newhandle()**

```
char** newhandle (
    long size )
```

Allocate relocatable memory.

Deprecated, use [sysmem\\_newhandle](#) instead

**Parameters**

<i>size</i>	The size to allocate in bytes.
-------------	--------------------------------

**Returns**

The allocated handle.

**See also**

[sysmem\\_newhandle\(\)](#)

### 38.23.4.8 sysmem\_copyptr()

```
void sysmem_copyptr (
    const void * src,
    void * dst,
    t_ptr_size bytes )
```

Copy memory the contents of one pointer to another pointer.

This function is similar to BlockMove() or memcpy(). It copies the contents of the memory from the source to the destination pointer.

**Parameters**

<i>src</i>	A pointer to the memory whose bytes will be copied.
<i>dst</i>	A pointer to the memory where the data will be copied.
<i>bytes</i>	The size in bytes of the data to be copied.

Referenced by jit\_copy\_bytes().

### 38.23.4.9 sysmem\_freehandle()

```
void sysmem_freehandle (
    t_handle handle )
```

Free memory allocated with [sysmem\\_newhandle\(\)](#).

**Parameters**

<i>handle</i>	The handle whose memory will be freed.
---------------	--

Referenced by jit\_handle\_free().

### 38.23.4.10 sysmem\_freeptr()

```
void sysmem_freeptr (
    void * ptr )
```

Free memory allocated with [sysmem\\_newptr\(\)](#).

This function is similar to DisposePtr or free. It frees the memory that had been allocated to the given pointer.

#### Parameters

<i>ptr</i>	The pointer whose memory will be freed.
------------	---

Referenced by [jit\\_disposeptr\(\)](#), and [jit\\_freebytes\(\)](#).

### 38.23.4.11 sysmem\_handlesize()

```
t_ptr_size sysmem_handlesize (
    t_handle handle )
```

Find the size of a handle.

This function is similar to GetHandleSize().

#### Parameters

<i>handle</i>	The handle whose size will be queried.
---------------	--

#### Returns

The number of bytes allocated to the specified handle.

Referenced by [jit\\_handle\\_size\\_get\(\)](#).

### 38.23.4.12 sysmem\_lockhandle()

```
int sysmem_lockhandle (
    t_handle handle,
    int lock )
```

Set the locked/unlocked state of a handle.

This function is similar to HLock or HUnlock. It sets the lock state of a handle, using a zero or non-zero number.

**Parameters**

<i>handle</i>	The handle that will be locked.
<i>lock</i>	The new lock state of the handle.

**Returns**

The previous lock state.

Referenced by `jit_handle_lock()`.

**38.23.4.13 sysmem\_newhandle()**

```
t_handle sysmem_newhandle (
    t_ptr_size size )
```

Allocate a handle (a pointer to a pointer).

This function is similar to `NewHandle()`. It allocates a handle of a given number of bytes and returns a [t\\_handle](#).

**Parameters**

<i>size</i>	The size of the handle in bytes that will be allocated.
-------------	---

**Returns**

A new [t\\_handle](#).

Referenced by `jit_handle_new()`.

**38.23.4.14 sysmem\_newhandleclear()**

```
t_handle sysmem_newhandleclear (
    t_ptr_size size )
```

Allocate a handle (a pointer to a pointer) whose memory is set to zero.

**Parameters**

<i>size</i>	The size of the handle in bytes that will be allocated.
-------------	---

**Returns**

A new [t\\_handle](#).

**See also**

[sysmem\\_newhandle\(\)](#)

### 38.23.4.15 sysmem\_newptr()

```
BEGIN_USING_C_LINKAGE t_ptr sysmem_newptr (
    t_ptr_size size )
```

Allocate memory.

This function is similar to NewPtr() or malloc(). It allocates a pointer of a given number of bytes and returns a pointer to the memory allocated.

**Parameters**

<i>size</i>	The amount of memory to allocate.
-------------	-----------------------------------

**Returns**

A pointer to the allocated memory, or NULL if the allocation fails.

Referenced by [jit\\_getbytes\(\)](#), and [jit\\_newptr\(\)](#).

### 38.23.4.16 sysmem\_newptrclear()

```
t_ptr sysmem_newptrclear (
    t_ptr_size size )
```

Allocate memory and set it to zero.

This function is similar to NewPtrClear() or calloc(). It allocates a pointer of a given number of bytes, zeroing all memory, and returns a pointer to the memory allocated.

**Parameters**

<i>size</i>	The amount of memory to allocate.
-------------	-----------------------------------

**Returns**

A pointer to the allocated memory, or NULL if the allocation fails.

**38.23.4.17 sysmem\_nullterminatehandle()**

```
t_max_err sysmem_nullterminatehandle (
    t_handle h )
```

Add a null terminator to a handle.

**Parameters**

<i>h</i>	A handle to null terminate.
----------	-----------------------------

**Returns**

An error code.

**38.23.4.18 sysmem\_ptrandhand()**

```
t_max_err sysmem_ptrandhand (
    void * p,
    t_handle h,
    t_ptr_size size )
```

Add memory to an existing handle and copy memory to the resized portion from a pointer.

This function is similar to PtrAndHand(). It resizes an existing handle by adding a given number of bytes to it and copies data from a pointer into those bytes.

**Parameters**

<i>p</i>	The existing pointer whose data will be copied into the resized handle.
<i>h</i>	The handle which will be enlarged by the size of the pointer.
<i>size</i>	The size in bytes that will be added to the handle.

**Returns**

The number of bytes allocated to the specified handle.

### 38.23.4.19 sysmem\_ptrbeforehand()

```
t_max_err sysmem_ptrbeforehand (
    void * p,
    t_handle h,
    t_ptr_size size )
```

Add memory to an existing handle and copy memory to the resized portion from a pointer.

Unlike [sysmem\\_ptrandhand\(\)](#), however, this copies the ptr before the previously existing handle data.

#### Parameters

<i>p</i>	The existing pointer whose data will be copied into the resized handle.
<i>h</i>	The handle which will be enlarged by the size of the pointer.
<i>size</i>	The size in bytes that will be added to the handle.

#### Returns

An error code.

### 38.23.4.20 sysmem\_ptrsize()

```
t_ptr_size sysmem_ptrsize (
    void * ptr )
```

Find the size of a pointer.

This function is similar to `_msize()`.

#### Parameters

<i>ptr</i>	The pointer whose size will be queried
------------	--

#### Returns

The number of bytes allocated to the pointer specified.

### 38.23.4.21 sysmem\_resizehandle()

```
t_max_err sysmem_resizehandle (
    t_handle handle,
    t_ptr_size newsize )
```

Resize an existing handle.

This function is similar to SetHandleSize(). It resizes an existing handle to the size specified.

#### Parameters

<i>handle</i>	The handle that will be resized.
<i>newsized</i>	The new size of the handle in bytes.

#### Returns

The number of bytes allocated to the specified handle.

Referenced by jit\_handle\_size\_set().

### 38.23.4.22 sysmem\_resizeptr()

```
t_ptr sysmem_resizeptr (
    void * ptr,
    t_ptr_size newsized )
```

Resize an existing pointer.

This function is similar to realloc(). It resizes an existing pointer and returns a new pointer to the resized memory.

#### Parameters

<i>ptr</i>	The pointer to the memory that will be resized.
<i>newsized</i>	The new size of the pointer in bytes.

#### Returns

A pointer to the resized memory, or NULL if the allocation fails.

### 38.23.4.23 sysmem\_resizeptrclear()

```
t_ptr sysmem_resizeptrclear (
    void * ptr,
    t_ptr_size newsized )
```

Resize an existing pointer and clear the newly allocated memory, if any.

**Parameters**

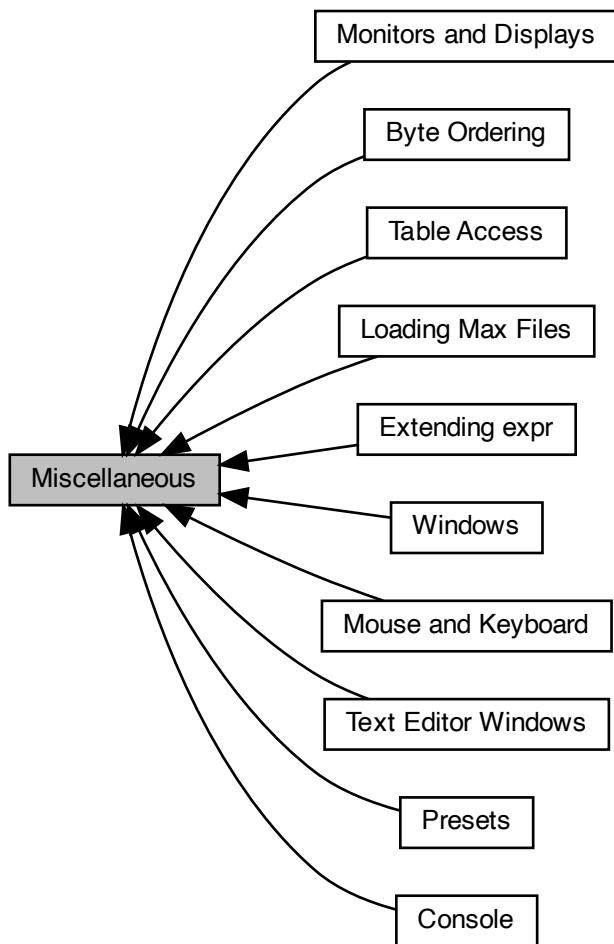
<i>ptr</i>	The pointer to the memory that will be resized.
<i>newsize</i>	The new size of the pointer in bytes.

**Returns**

A pointer to the resized memory, or NULL if the allocation fails.

## 38.24 Miscellaneous

Collaboration diagram for Miscellaneous:



## Modules

- [Console](#)
- [Byte Ordering](#)

*Utilities for swapping the order of bytes to match the Endianness of the required platform.*

- [Extending expr](#)

*If you want to use C-like variable expressions that are entered by a user of your object, you can use the "guts" of Max's expr object in your object.*

- [Table Access](#)

*You can use these functions to access named table objects.*

- [Text Editor Windows](#)

*Max has a simple built-in text editor object that can display and edit text in conjunction with your object.*

- [Presets](#)

*Max contains a preset object that has the ability to send preset messages to some or all of the objects (clients) in a Patcher window.*

- [Loading Max Files](#)

*Several high-level functions permit you to load patcher files.*

- [Monitors and Displays](#)

*Functions for finding our information about the environment.*

- [Windows](#)

- [Mouse and Keyboard](#)

## Macros

- [#define INRANGE\(v, lo, hi\)](#)

*If a value is within the specified range, then return true.*

- [#define MAX\(a, b\)](#)

*Return the higher of two values.*

- [#define MIN\(a, b\)](#)

*Return the lower of two values.*

- [#define CLAMP\(a, lo, hi\)](#)

*Limit values to within a specified range.*

- [#define BEGIN\\_USING\\_C\\_LINKAGE](#)

*Ensure that any definitions following this macro use a C-linkage, not a C++ linkage.*

- [#define END\\_USING\\_C\\_LINKAGE](#)

*Close a definition section that was opened using [BEGIN\\_USING\\_C\\_LINKAGE](#).*

- [#define calcoffset\(x, y\)](#)

*Find byte offset of a named member of a struct, relative to the beginning of that struct.*

- [#define structmembersize\(structname, membername\)](#)

*Find size of a named member of a struct.*

## Typedefs

- `typedef unsigned int t_uint`  
`an unsigned int as defined by the architecture / platform`
- `typedef int8_t t_int8`  
`a 1-byte int`
- `typedef uint8_t t_uint8`  
`an unsigned 1-byte int`
- `typedef int16_t t_int16`  
`a 2-byte int`
- `typedef uint16_t t_uint16`  
`an unsigned 2-byte int`
- `typedef int32_t t_int32`  
`a 4-byte int`
- `typedef uint32_t t_uint32`  
`an unsigned 4-byte int`
- `typedef int64_t t_int64`  
`an 8-byte int`
- `typedef uint64_t t_uint64`  
`an unsigned 8-byte int`
- `typedef t_uint32 t_fourcc`  
`an integer of suitable size to hold a four char code / identifier`
- `typedef uintptr_t t_ptr_uint`  
`an unsigned pointer-sized int`
- `typedef intptr_t t_ptr_int`  
`a pointer-sized int`
- `typedef float t_atom_float`  
`the type that is an A_FLOAT in a t_atom`
- `typedef short t_getbytes_size`  
`like size_t but for getbytes()`
- `typedef t_ptr_int t_int`  
`an integer`
- `typedef t_ptr_uint t_ptr_size`  
`unsigned pointer-sized value for counting (like size_t)`
- `typedef t_ptr_int t_atom_long`  
`the type that is an A_LONG in a t_atom`
- `typedef t_atom_long t_max_err`  
`an integer value suitable to be returned as an error code`
- `typedef char ** t_handle`  
`a handle (address of a pointer)`
- `typedef char * t_ptr`  
`a pointer`
- `typedef t_uint8 t_bool`  
`a true/false variable`
- `typedef t_int16 t_filepath`  
`i.e. path/vol in file APIs identifying a folder`

## Enumerations

- enum `e_max_wind_advise_result` { `aaYes` , `aaNo` , `aaCancel` }

*Returned values from `wind_advise()`*

- enum `e_max_errorcodes` {  
`MAX_ERR_NONE` , `MAX_ERR_GENERIC` , `MAX_ERR_INVALID_PTR` , `MAX_ERR_DUPLICATE` ,  
`MAX_ERR_OUT_OF_MEM` }

*Standard values returned by function calls with a return type of `t_max_err`.*

## Functions

- `BEGIN_USING_C_LINKAGE void * globalsymbol_reference (t_object *x, C74_CONST char *name, C74_CONST char *classname)`  
*Get a reference to an object that is bound to a `t_symbol`.*
- `void globalsymbol_dereference (t_object *x, C74_CONST char *name, C74_CONST char *classname)`  
*Stop referencing an object that is bound to a `t_symbol`, previously referenced using `globalsymbol_reference()`.*
- `t_max_err globalsymbol_bind (t_object *x, C74_CONST char *name, long flags)`  
*Bind an object to a `t_symbol`.*
- `void globalsymbol_unbind (t_object *x, C74_CONST char *name, long flags)`  
*Remove an object from being bound to a `t_symbol`.*
- `t_atom_long method_true (void *x)`  
*A method that always returns true.*
- `t_atom_long method_false (void *x)`  
*A method that always returns false.*
- `t_symbol * symbol_unique (void)`  
*Generates a unique `t_symbol` \*.*
- `t_symbol * symbol_stripquotes (t_symbol *s)`  
*Strip quotes from the beginning and end of a symbol if they are present.*
- `void error_sym (void *x, t_symbol *s)`  
*Posts an error message to the Max window.*
- `void post_sym (void *x, t_symbol *s)`  
*Posts a message to the Max window.*
- `t_max_err symbolarray_sort (long ac, t_symbol **av)`  
*Performs an ASCII sort on an array of `t_symbol` \*s.*
- `void object_obex_quickref (void *x, long *numitems, t_symbol **items)`  
*Developers do not need to directly use the `object_obex_quickref()` function.*
- `void error_subscribe (t_object *x)`  
*Receive messages from the error handler.*
- `void error_unsubscribe (t_object *x)`  
*Remove an object as an error message recipient.*
- `void quittask_install (method m, void *a)`  
*Register a function that will be called when Max exits.*
- `void quittask_remove (method m)`  
*Unregister a function previously registered with `quittask_install()`.*
- `short maxversion (void)`  
*Determine version information about the current Max environment.*
- `BEGIN_USING_C_LINKAGE char * strncpy_zero (char *dst, const char *src, long size)`

*Copy the contents of one string to another, in a manner safer than the standard strcpy() or strncpy().*

- `char * strncat_zero (char *dst, const char *src, long size)`

*Concatenate the contents of one string onto the end of another, in a manner safer than the standard strcat() or strncat().*

- `int sprintf_zero (char *buffer, size_t count, const char *format,...)`

*Copy the contents of a string together with value substitutions, in a manner safer than the standard sprintf() or snprintf().*

- `short wind_advise (t_object *w, const char *s,...)`

*Throw a dialog which may have text and up to three buttons.*

- `void wind_setcursor (short which)`

*Change the cursor.*

### 38.24.1 Detailed Description

### 38.24.2 Macro Definition Documentation

#### 38.24.2.1 BEGIN\_USING\_C\_LINKAGE

```
#define BEGIN_USING_C_LINKAGE
```

Ensure that any definitions following this macro use a C-linkage, not a C++ linkage.

The Max API uses C-linkage. This is important for objects written in C++ or that use a C++ compiler. This macro must be balanced with the `END_USING_C_LINKAGE` macro.

#### 38.24.2.2 calcoffset

```
#define calcoffset(
    x,
    y )
```

Find byte offset of a named member of a struct, relative to the beginning of that struct.

##### Parameters

x	The name of the struct
y	The name of the member

##### Returns

A pointer-sized integer representing the number of bytes into the struct where the member begins.

### 38.24.2.3 CLAMP

```
#define CLAMP (
    a,
    lo,
    hi )
```

Limit values to within a specified range.

#### Parameters

<i>a</i>	The value to constrain. NB: CLIP_ASSIGN modifies param 'a' but CLAMP only returns limited value
<i>lo</i>	The low bound for the range.
<i>hi</i>	The high bound for the range.

#### Returns

Returns the value *a* constrained to the range specified by *lo* and *hi*.

### 38.24.2.4 INRANGE

```
#define INRANGE (
    v,
    lo,
    hi )
```

If a value is within the specified range, then return true.

Otherwise return false.

#### Parameters

<i>v</i>	The value to test.
<i>lo</i>	The low bound for the range.
<i>hi</i>	The high bound for the range.

#### Returns

Returns true if within range, otherwise false.

### 38.24.2.5 MAX

```
#define MAX(  
    a,  
    b )
```

Return the higher of two values.

#### Parameters

a	The first value to compare.
b	The second value to compare.

#### Returns

Returns the higher of a or b.

### 38.24.2.6 MIN

```
#define MIN(  
    a,  
    b )
```

Return the lower of two values.

#### Parameters

a	The first value to compare.
b	The second value to compare.

#### Returns

Returns the lower of a or b.

### 38.24.2.7 structmembersize

```
#define structmembersize(  
    structname,  
    membername )
```

Find size of a named member of a struct.

**Parameters**

<i>structname</i>	The name of the struct
<i>membername</i>	The name of the member

**Returns**

The size of the member of the struct.

### 38.24.3 Enumeration Type Documentation

#### 38.24.3.1 e\_max\_errorcodes

enum [e\\_max\\_errorcodes](#)

Standard values returned by function calls with a return type of [t\\_max\\_err](#).

**Enumerator**

MAX_ERR_NONE	No error.
MAX_ERR_GENERIC	Generic error.
MAX_ERR_INVALID_PTR	Invalid Pointer.
MAX_ERR_DUPLICATE	Duplicate.
MAX_ERR_OUT_OF_MEM	Out of memory.

#### 38.24.3.2 e\_max\_wind\_advise\_result

enum [e\\_max\\_wind\\_advise\\_result](#)

Returned values from [wind\\_advise\(\)](#)

**Enumerator**

aaYes	Yes button was chosen.
aaNo	No button was chosen.
aaCancel	Cancel button was chosen.

## 38.24.4 Function Documentation

### 38.24.4.1 error\_subscribe()

```
void error_subscribe (
    t_object * x )
```

Receive messages from the error handler.

#### Parameters

x	The object to be subscribed to the error handler.
---	---

#### Remarks

`error_subscribe()` enables your object to receive a message (error), followed by the list of atoms in the error message posted to the Max window.

Prior to calling `error_subscribe()`, you should bind the error message to an internal error handling routine:

```
addmess((method)myobject_error, "error", A_GIMME, 0);
```

Your error handling routine should be declared as follows:

```
void myobject_error(t_myobject *x, t_symbol *s, short argc, t_atom *argv);
```

### 38.24.4.2 error\_sym()

```
void error_sym (
    void * x,
    t_symbol * s )
```

Posts an error message to the Max window.

This function is interrupt safe.

#### Parameters

x	The object's pointer
s	Symbol to be posted as an error in the Max window

### 38.24.4.3 error\_unsubscribe()

```
void error_unsubscribe (
    t_object * x )
```

Remove an object as an error message recipient.

#### Parameters

x	The object to unsubscribe.
---	----------------------------

### 38.24.4.4 globalsymbol\_bind()

```
t_max_err globalsymbol_bind (
    t_object * x,
    C74_CONST char * name,
    long flags )
```

Bind an object to a [t\\_symbol](#).

#### Parameters

x	The object to bind to the <a href="#">t_symbol</a> .
name	The name of the <a href="#">t_symbol</a> to which the object will be bound.
flags	Pass 0.

#### Returns

A Max error code.

### 38.24.4.5 globalsymbol\_dereference()

```
void globalsymbol_dereference (
    t_object * x,
    C74_CONST char * name,
    C74_CONST char * classname )
```

Stop referencing an object that is bound to a [t\\_symbol](#), previously referenced using [globalsymbol\\_reference\(\)](#).

#### Parameters

x	The object that is getting the reference to the symbol.
name	The name of the symbol to reference.
classname	The name of the class of which the object we are referencing should be an instance.

## See also

[globalsymbol\\_reference\(\)](#)

### 38.24.4.6 `globalsymbol_reference()`

```
BEGIN_USING_C_LINKAGE void* globalsymbol_reference (
    t_object * x,
    C74_CONST char * name,
    C74_CONST char * classname )
```

Get a reference to an object that is bound to a [t\\_symbol](#).

## Parameters

<i>x</i>	The object that is getting the reference to the symbol.
<i>name</i>	The name of the symbol to reference.
<i>classname</i>	The name of the class of which the object we are referencing should be an instance.

## Returns

The *s\_thing* of the [t\\_symbol](#).

## Remarks

An example of real-world use is to get the buffer~ object associated with a symbol.

```
// the struct of our object
typedef struct _myobject {
    t_object     obj;
    t_symbol     *buffer_name;
    t_buffer     *buffer_object;
} t_myobject;
void myobject_setbuffer(t_myobject *x, t_symbol *s, long argc, t_atom *argv)
{
    if(s != x->buffer_name) {
        // Reference the buffer associated with the incoming name
        x->buffer_object = (t_buffer *)globalsymbol_reference((t_object *)x, s->s_name, "buffer~");

        // If we were previously referencing another buffer, we should not longer reference it.
        globalsymbol_dereference((t_object *)x, x->buffer_name->s_name, "buffer~");
        x->buffer_name = s;
    }
}
```

### 38.24.4.7 `globalsymbol_unbind()`

```
void globalsymbol_unbind (
    t_object * x,
    C74_CONST char * name,
    long flags )
```

Remove an object from being bound to a [t\\_symbol](#).

### Parameters

<i>x</i>	The object from which to unbind the <a href="#">t_symbol</a> .
<i>name</i>	The name of the <a href="#">t_symbol</a> from which the object will be unbound.
<i>flags</i>	Pass 0.

### 38.24.4.8 maxversion()

```
short maxversion (
    void )
```

Determine version information about the current Max environment.

This function returns the version number of Max. In Max versions 2.1.4 and later, this number is the version number of the Max kernel application in binary-coded decimal. Thus, 2.1.4 would return 214 hex or 532 decimal. Version 3.0 returns 300 hex.

Use this to check for the existence of particular function macros that are only present in more recent Max versions. Versions before 2.1.4 returned 1, except for versions 2.1.1 - 2.1.3 which returned 2.

Bit 14 (counting from left) will be set if Max is running as a standalone application, so you should mask the lower 12 bits to get the version number.

### Returns

The Max environment's version number.

### 38.24.4.9 object\_obex\_quickref()

```
void object_obex_quickref (
    void * x,
    long * numitems,
    t_symbol ** items )
```

Developers do not need to directly use the [object\\_obex\\_quickref\(\)](#) function.

It was used in Max 4 to add support for attributes to the quickref, but this is automatic in Max 5.

Referenced by [ext\\_main\(\)](#).

### 38.24.4.10 post\_sym()

```
void post_sym (
    void * x,
    t_symbol * s )
```

Posts a message to the Max window.

This function is interrupt safe.

**Parameters**

<i>x</i>	The object's pointer
<i>s</i>	Symbol to be posted in the Max window

**38.24.4.11 quittask\_install()**

```
void quittask_install (
    method m,
    void * a )
```

Register a function that will be called when Max exits.

**Parameters**

<i>m</i>	A function that will be called on Max exit.
<i>a</i>	Argument to be used with method <i>m</i> .

**Remarks**

[quittask\\_install\(\)](#) provides a mechanism for your external to register a routine to be called prior to Max shutdown. This is useful for objects that need to provide disk-based persistency outside the standard Max storage mechanisms, or need to shut down hardware or their connection to system software and cannot do so in the termination routine of a code fragment.

**38.24.4.12 quittask\_remove()**

```
void quittask_remove (
    method m )
```

Unregister a function previously registered with [quittask\\_install\(\)](#).

**Parameters**

<i>m</i>	Function to be removed as a shutdown method.
----------	--

**38.24.4.13 snprintf\_zero()**

```
int snprintf_zero (
```

```
char * buffer,
size_t count,
const char * format,
... )
```

Copy the contents of a string together with value substitutions, in a manner safer than the standard sprintf() or snprintf().

This is the preferred function to use for this operation in Max.

#### Parameters

<i>buffer</i>	The destination string (already allocated) for the copy.
<i>count</i>	The number of chars allocated to the buffer string.
<i>format</i>	The source string that will be copied, which may include sprintf() formatting codes for substitutions.
...	An array of arguments to be substituted into the format string.

Referenced by max\_jit\_mop\_assist().

#### 38.24.4.14 strncat\_zero()

```
char* strncat_zero (
    char * dst,
    const char * src,
    long size )
```

Concatenate the contents of one string onto the end of another, in a manner safer than the standard strcat() or strncat().

This is the preferred function to use for this operation in Max.

#### Parameters

<i>dst</i>	The destination string onto whose end the src string will be appended.
<i>src</i>	The source string that will be copied.
<i>size</i>	The number of chars allocated to the dst string.

#### 38.24.4.15 strncpy\_zero()

```
BEGIN_USING_C_LINKAGE char* strncpy_zero (
    char * dst,
    const char * src,
    long size )
```

Copy the contents of one string to another, in a manner safer than the standard strcpy() or strncpy().

This is the preferred function to use for this operation in Max.

**Parameters**

<i>dst</i>	The destination string (already allocated) for the copy.
<i>src</i>	The source string that will be copied.
<i>size</i>	The number of chars allocated to the dst string.

**38.24.4.16 symbol\_stripquotes()**

```
t_symbol* symbol_stripquotes (
    t_symbol * s )
```

Strip quotes from the beginning and end of a symbol if they are present.

**Parameters**

<i>s</i>	The symbol to be stripped.
----------	----------------------------

**Returns**

Symbol with any leading/trailing quote pairs removed.

**38.24.4.17 symbol\_unique()**

```
t_symbol* symbol_unique (
    void )
```

Generates a unique [t\\_symbol](#) \*.

The symbol will be formatted somewhat like "u123456789".

**Returns**

This function returns a unique [t\\_symbol](#) \*.

**38.24.4.18 symbolarray\_sort()**

```
t_max_err symbolarray_sort (
    long ac,
    t_symbol ** av )
```

Performs an ASCII sort on an array of [t\\_symbol](#) \*s.

**Parameters**

<i>ac</i>	The count of <a href="#">t_symbol</a> *s in <i>av</i>
<i>av</i>	An array of <a href="#">t_symbol</a> *s to be sorted

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

Referenced by [jit\\_object\\_exportattrs\(\)](#).

**38.24.4.19 wind\_advise()**

```
short wind_advise (
    t\_object * w,
    const char * s,
    ... )
```

Throw a dialog which may have text and up to three buttons.

For example, this can be used to ask "Save changes before..."

**Parameters**

<i>w</i>	The window with which this dialog is associated.
<i>s</i>	A string with any sprintf()-like formatting to be displayed.
...	Any variables that should be substituted in the string defined by <i>s</i> .

**Returns**

One of the values defined in [e\\_max\\_wind\\_advise\\_result](#), depending on what the user selected.

**38.24.4.20 wind\_setcursor()**

```
void wind_setcursor (
    short which )
```

Change the cursor.

### Parameters

<code>which</code>	One of the following predefined cursors: <pre>#define C_ARROW 1 #define C_WATCH 2 #define C_IBEAM 3 #define C_HAND 4 #define C_CROSS 5 #define C_PENCIL 6 #define C_GROW 8</pre>
--------------------	---

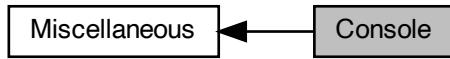
### Remarks

`wind_setcursor()` keeps track of what the cursor was previously set to, so if something else has changed the cursor, you may not see a new cursor if you set it to the previous argument to `wind_setcursor()`.

The solution is to call `wind_setcursor(0)` before calling it with the desired cursor constant. Use `wind_setcursor(-1)` to tell Max you'll set the cursor to your own cursor directly.

## 38.25 Console

Collaboration diagram for Console:



### Functions

- **BEGIN\_USING\_C\_LINKAGE void `post` (C74\_CONST char \*fmt,...)**  
*Print text to the Max window.*
- **void `cpost` (C74\_CONST char \*fmt,...)**  
*Print text to the system console.*
- **void `error` (C74\_CONST char \*fmt,...)**  
*Print an error to the Max window.*
- **void `ouchstring` (C74\_CONST char \*s,...)**  
*Put up an error or advisory alert box on the screen.*
- **void `postatom` (`t_atom` \*ap)**  
*Print multiple items in the same line of text in the Max window.*
- **void `object_post` (`t_object` \*x, C74\_CONST char \*s,...)**  
*Print text to the Max window, linked to an instance of your object.*

- void [object\\_error](#) ([t\\_object](#) \*x, C74\_CONST char \*s,...)  
*Print text to the Max window, linked to an instance of your object, and flagged as an error (highlighted with a red background).*
- void [object\\_warn](#) ([t\\_object](#) \*x, C74\_CONST char \*s,...)  
*Print text to the Max window, linked to an instance of your object, and flagged as a warning (highlighted with a yellow background).*
- void [object\\_error\\_obtrusive](#) ([t\\_object](#) \*x, C74\_CONST char \*s,...)  
*Print text to the Max window, linked to an instance of your object, and flagged as an error (highlighted with a red background), and grab the user's attention by displaying a banner in the patcher window.*

### 38.25.1 Detailed Description

### 38.25.2 Function Documentation

#### 38.25.2.1 cpost()

```
void cpost (
    C74_CONST char * fmt,
    ...
)
```

Print text to the system console.

On the Mac this post will be visible by launching Console.app in the /Applications/Utilities folder. On Windows this post will be visible by launching the dbgView.exe program, which is a free download as a part of Microsoft's SysInternals.

#### Parameters

<i>fmt</i>	A C-string containing text and printf-like codes specifying the sizes and formatting of the additional arguments.
...	Arguments of any type that correspond to the format codes in <i>fmtString</i> .

#### Remarks

Particularly on MacOS 10.5, posting to Console.app can be a computationally expensive operation. Use with care.

#### See also

[post\(\)](#)  
[object\\_post\(\)](#)

### 38.25.2.2 `error()`

```
void error (
    C74_CONST char * fmt,
    ...
)
```

Print an error to the Max window.

Max 5 introduced `object_error()`, which provides several enhancements to `error()` where a valid `t_object` pointer is available.

`error()` is very similar to `post()`, thought it offers two additional features:

- the post to the Max window is highlighted (with a red background).
- the post can be trapped with the error object in a patcher.

#### Parameters

<code>fmt</code>	A C-string containing text and printf-like codes specifying the sizes and formatting of the additional arguments.
<code>...</code>	Arguments of any type that correspond to the format codes in <code>fmtString</code> .

#### See also

[object\\_post\(\)](#)  
[post\(\)](#)  
[cpost\(\)](#)

### 38.25.2.3 `object_error()`

```
void object_error (
    t_object * x,
    C74_CONST char * s,
    ...
)
```

Print text to the Max window, linked to an instance of your object, and flagged as an error (highlighted with a red background).

Max window rows which are generated using `object_post()` or `object_error()` can be double-clicked by the user to have Max assist with locating the object in a patcher. Rows created with `object_post()` and `object_error()` will also automatically provide the name of the object's class in the correct column in the Max window.

#### Parameters

<code>x</code>	A pointer to your object.
<code>s</code>	A C-string containing text and printf-like codes specifying the sizes and formatting of the additional arguments.
<code>...</code>	Arguments of any type that correspond to the format codes in <code>fmtString</code> .

**See also**

[object\\_post\(\)](#)  
[object\\_warn\(\)](#)

**38.25.2.4 object\_error\_obtrusive()**

```
void object_error_obtrusive (
    t_object * x,
    C74_CONST char * s,
    ... )
```

Print text to the Max window, linked to an instance of your object, and flagged as an error (highlighted with a red background), and grab the user's attention by displaying a banner in the patcher window.

This function should be used exceedingly sparingly, with preference given to [object\\_error\(\)](#) when a problem occurs.

**Parameters**

x	A pointer to your object.
s	A C-string containing text and printf-like codes specifying the sizes and formatting of the additional arguments.
...	Arguments of any type that correspond to the format codes in fmtString.

**See also**

[object\\_post\(\)](#)  
[object\\_error\(\)](#)

**38.25.2.5 object\_post()**

```
void object_post (
    t_object * x,
    C74_CONST char * s,
    ... )
```

Print text to the Max window, linked to an instance of your object.

Max window rows which are generated using [object\\_post\(\)](#) or [object\\_error\(\)](#) can be double-clicked by the user to have Max assist with locating the object in a patcher. Rows created with [object\\_post\(\)](#) and [object\\_error\(\)](#) will also automatically provide the name of the object's class in the correct column in the Max window.

### Parameters

<code>x</code>	A pointer to your object.
<code>s</code>	A C-string containing text and printf-like codes specifying the sizes and formatting of the additional arguments.
...	Arguments of any type that correspond to the format codes in fmtString.

### Remarks

**Example:**

```
void myMethod(myObject *x, long someArgument)
{
    object_post((t_object*)x, "This is my argument: %ld", someArgument);
}
```

### See also

[object\\_error\(\)](#)

### 38.25.2.6 `object_warn()`

```
void object_warn (
    t_object * x,
    C74_CONST char * s,
    ...
)
```

Print text to the Max window, linked to an instance of your object, and flagged as a warning (highlighted with a yellow background).

Max window rows which are generated using `object_post()`, `object_error()`, or `object_warn` can be double-clicked by the user to have Max assist with locating the object in a patcher. Rows created with `object_post()`, `object_error()`, or `object_warn()` will also automatically provide the name of the object's class in the correct column in the Max window.

### Parameters

<code>x</code>	A pointer to your object.
<code>s</code>	A C-string containing text and printf-like codes specifying the sizes and formatting of the additional arguments.
...	Arguments of any type that correspond to the format codes in fmtString.

### See also

[object\\_post\(\)](#)  
[object\\_error\(\)](#)

### 38.25.2.7 `ouchstring()`

```
void ouchstring (
    C74_CONST char * s,
    ...
)
```

Put up an error or advisory alert box on the screen.

Don't use this function. Instead use [error\(\)](#), [object\\_error\(\)](#), or [object\\_error\\_obtrusive\(\)](#).

This function performs an `sprintf()` on `fmtString` and items, then puts up an alert box. `ouchstring()` will queue the message to a lower priority level if it's called in an interrupt and there is no alert box request already pending.

#### Parameters

<code>s</code>	A C-string containing text and printf-like codes specifying the sizes and formatting of the additional arguments.
...	Arguments of any type that correspond to the format codes in <code>fmtString</code> .

#### See also

[error\(\)](#)  
[object\\_error\(\)](#)  
[object\\_error\\_obtrusive\(\)](#)

### 38.25.2.8 `post()`

```
BEGIN_USING_C_LINKAGE void post (
    C74_CONST char * fmt,
    ...
)
```

Print text to the Max window.

Max 5 introduced [object\\_post\(\)](#), which provides several enhancements to [post\(\)](#) where a valid `t_object` pointer is available.

`post()` is a `printf()` for the Max window. It even works from non-main threads, queuing up multiple lines of text to be printed when the main thread processing resumes. [post\(\)](#) can be quite useful in debugging your external object.

#### Parameters

<code>fmt</code>	A C-string containing text and printf-like codes specifying the sizes and formatting of the additional arguments.
...	Arguments of any type that correspond to the format codes in <code>fmtString</code> .

### Remarks

Note that post only passes 16 bytes of arguments to sprintf, so if you want additional formatted items on a single line, use [postatom\(\)](#).

### Example:

```
short whatIsIt;
whatIsIt = 999;
post ("the variable is %ld", (long)whatIsIt);
```

### Remarks

The Max Window output when this code is executed.  
the variable is 999

### See also

[object\\_post\(\)](#)  
[error\(\)](#)  
[cpost\(\)](#)

## 38.25.2.9 postatom()

```
void postatom (
    t_atom * ap )
```

Print multiple items in the same line of text in the Max window.

This function prints a single [t\\_atom](#) on a line in the Max window without a carriage return afterwards, as [post\(\)](#) does. Each [t\\_atom](#) printed is followed by a space character.

### Parameters

<i>ap</i>	The address of a <a href="#">t_atom</a> to print.
-----------	---

### See also

[object\\_post\(\)](#)  
[post\(\)](#)  
[cpost\(\)](#)

## 38.26 Byte Ordering

Utilities for swapping the order of bytes to match the Endianness of the required platform.

Collaboration diagram for Byte Ordering:



## Macros

- `#define BYTEORDER_SWAPW16(x)`  
*Switch the byte ordering of a short integer.*
- `#define BYTEORDER_SWAPW32(x)`  
*Switch the byte ordering of an integer.*
- `#define BYTEORDER_SWAPW64(x)`  
*Switch the byte ordering of an integer.*
- `#define BYTEORDER_SWAPF32`  
*Switch the byte ordering of a float.*
- `#define BYTEORDER_SWAPF64`  
*Switch the byte ordering of a double.*

### 38.26.1 Detailed Description

Utilities for swapping the order of bytes to match the Endianness of the required platform.

An introduction to the issue of endianness can be found at <http://en.wikipedia.org/wiki/Endianness>.

Of particular relevance is that a Macintosh with a PPC processor uses a Big-endian byte ordering, whereas an Intel processor in a Mac or Windows machine will use a Little-endian byte ordering.

These utilities are defined to assist with cases where byte ordering needs to be manipulated for floats or ints. Note that floats are subject to the same byte ordering rules as integers. While the IEEE defines the bits, the machine still defines how the bits are arranged with regard to bytes.

### 38.26.2 Macro Definition Documentation

#### 38.26.2.1 BYTEORDER\_SWAPF32

```
#define BYTEORDER_SWAPF32
```

Switch the byte ordering of a float.

**Parameters**

x	A float.
---	----------

**Returns**

A float with the byte-ordering swapped.

**38.26.2.2 BYTEORDER\_SWAPF64**

```
#define BYTEORDER_SWAPF64
```

Switch the byte ordering of a double.

**Parameters**

x	A double.
---	-----------

**Returns**

A double.

**38.26.2.3 BYTEORDER\_SWAPW16**

```
#define BYTEORDER_SWAPW16( x )
```

Switch the byte ordering of a short integer.

**Parameters**

x	A short integer.
---	------------------

**Returns**

A short integer with the byte-ordering swapped.

### 38.26.2.4 BYTEORDER\_SWAPW32

```
#define BYTEORDER_SWAPW32 (  
    x )
```

Switch the byte ordering of an integer.

#### Parameters

x	An integer.
---	-------------

#### Returns

An integer with the byte-ordering swapped.

### 38.26.2.5 BYTEORDER\_SWAPW64

```
#define BYTEORDER_SWAPW64 (  
    x )
```

Switch the byte ordering of an integer.

#### Parameters

x	An integer.
---	-------------

#### Returns

An integer with the byte-ordering swapped.

## 38.27 Extending expr

If you want to use C-like variable expressions that are entered by a user of your object, you can use the "guts" of Max's expr object in your object.

Collaboration diagram for Extending expr:



## Data Structures

- struct `t_ex_ex`  
`ex_ex.`
- struct `t_expr`  
*Struct for an instance of expr.*

## Enumerations

- enum `e_max_expr_types` {
 `ET_INT` , `ET_FLT` , `ET_OP` , `ET_STR` ,
 `ET_TBL` , `ET_FUNC` , `ET_SYM` , `ET_VSYM` ,
 `ET_LP` , `ET_LB` , `ET_II` , `ET_FI` ,
 `ET_SI` }

*Defines for ex\_type.*

## Functions

- void \* `expr_new` (long argc, `t_atom` \*argv, `t_atom` \*types)  
*Create a new expr object.*
- short `expr_eval` (`t_expr` \*x, long argc, `t_atom` \*argv, `t_atom` \*result)  
*Evaluate an expression in an expr object.*

### 38.27.1 Detailed Description

If you want to use C-like variable expressions that are entered by a user of your object, you can use the "guts" of Max's expr object in your object.

For example, the if object uses expr routines for evaluating a conditional expression, so it can decide whether to send the message after the words then or else. The following functions provide an interface to expr.

### 38.27.2 Enumeration Type Documentation

#### 38.27.2.1 `e_max_expr_types`

```
enum e_max_expr_types
```

Defines for ex\_type.

We treat parenthesis and brackets special to keep a pointer to their match in the content.

## Enumerator

ET_INT	an int
ET_FLT	a float
ET_OP	operator
ET_STR	string
ET_TBL	a table, the content is a pointer
ET_FUNC	a function
ET_SYM	symbol ("string")
ET_VSYM	variable symbol ("\$s?")
ET_LP	left parenthesis
ET_LB	left bracket
ET_II	and integer inlet
ET_FI	float inlet
ET_SI	string inlet

## 38.27.3 Function Documentation

## 38.27.3.1 expr\_eval()

```
short expr_eval (
    t_expr * x,
    long argc,
    t_atom * argv,
    t_atom * result )
```

Evaluate an expression in an expr object.

## Parameters

x	The expr object to evaluate.
argc	Count of arguments in argv.
argv	Array of nine t_atom values that will be substituted for variable arguments (such as \$i1) in the expression. Unused arguments should be of type A_NOHING.
result	A pre-existing t_atom that will hold the type and value of the result of evaluating the expression.

## Returns

## Remarks

Evaluates the expression in an expr object with arguments in argv and returns the type and value of the evaluated expression as a `t_atom` in result. result need only point to a single `t_atom`, but argv should contain at least argc `t_atom` values. If, as in the example shown above under `expr_new()`, there are “gaps” between arguments, they should be filled in with `t_atom` of type `A NOTHING`.

### 38.27.3.2 `expr_new()`

```
void* expr_new (
    long argc,
    t_atom * argv,
    t_atom * types )
```

Create a new expr object.

## Parameters

<code>argc</code>	Count of arguments in argv.
<code>argv</code>	Arguments that are used to create the expr. See the example below for details.
<code>types</code>	A pre-existing array of nine <code>t_atoms</code> , that will hold the types of any variable arguments created in the expr. The types are returned in the <code>a_type</code> field of each <code>t_atom</code> . If an argument was not present, <code>A NOTHING</code> is returned.

## Returns

`expr_new()` creates an expr object from the arguments in argv and returns the type of any expr-style arguments contained in argv (i.e. \$i1, etc.) in atoms in an array pointed to by types.

## Remarks

types should already exist as an array of nine `t_atom` values, all of which will be filled in by `expr_new()`. If an argument was not present, it will set to type `A NOTHING`. For example, suppose argv pointed to the following atoms:

```
$i1 (A_SYM)
+
$ f3 (A_SYM)
+
3 (A_LONG)
```

After calling `expr_new`, types would contain the following:

Index	Argument	Type	Value
0	1 (\$i1)	A_LONG	0
1	2	A NOTHING	0
2	3 (\$f3)	A_FLOAT	0.0
3	4	A NOTHING	0
4	5	A NOTHING	0
5	6	A NOTHING	0
6	7	A NOTHING	0
7	8	A NOTHING	0
8	9	A NOTHING	0

## 38.28 Table Access

You can use these functions to access named table objects.

Collaboration diagram for Table Access:



### Functions

- short `table_get (t_symbol *s, long ***hp, long *sp)`  
*Get a handle to the data in a named table object.*
- short `table_dirty (t_symbol *s)`  
*Mark a table object as having changed data.*

### 38.28.1 Detailed Description

You can use these functions to access named table objects.

Tables have names when the user creates a table with an argument.

The scenario for knowing the name of a table but not the object itself is if you were passed a `t_symbol`, either as an argument to your creation function or in some message, with the implication being "do your thing with the data in the table named norris."

### 38.28.2 Function Documentation

#### 38.28.2.1 `table_dirty()`

```
short table_dirty (
    t_symbol * s )
```

Mark a table object as having changed data.

**Parameters**

<b>s</b>	Symbol containing the name of a table object.
----------	---

**Returns**

If no table is associated with tableName, table\_dirty returns a non-zero result.

**38.28.2.2 table\_get()**

```
short table_get (
    t_symbol * s,
    long *** hp,
    long * sp )
```

Get a handle to the data in a named table object.

**Parameters**

<b>s</b>	Symbol containing the name of the table object to find.
<b>hp</b>	Address of a handle where the table's data will be returned if the named table object is found.
<b>sp</b>	Number of elements in the table (its size in longs).

**Returns**

If no table object is associated with the symbol tableName, [table\\_get\(\)](#) returns a non-zero result.

**Remarks**

table\_get searches for a table associated with the [t\\_symbol](#) tableName. If one is found, a Handle to its elements (stored as an array of long integers) is returned and the function returns 0. Never count on a table to exist across calls to one of your methods. Call table\_get and check the result each time you wish to use a table.

Here is an example of retrieving the 40th element of a table:

```
long **storage,size,value;
if (!table_get(gensym("somename"),&storage,&size)) {
    if (size > 40)
        value = *((*storage)+40);
}
```

**38.29 Text Editor Windows**

Max has a simple built-in text editor object that can display and edit text in conjunction with your object.

Collaboration diagram for Text Editor Windows:



Max has a simple built-in text editor object that can display and edit text in conjunction with your object.

The routines described here let you create a text editor.

When the editor window is about to be closed, your object could receive as many as three messages. The first one, okclose, will be sent if the user has changed the text in the window. This is the standard okclose message that is sent to all "dirty" windows when they are about to be closed, but the text editor window object passes it on to you instead of doing anything itself. Refer to the section on Window Messages for a description of how to write a method for the okclose message. It's not required that you write one—if you don't, the behavior of the window will be determined by the setting of the window's w\_scratch bit. If it's set, no confirmation will be asked when a dirty window is closed (and no okclose message will be sent to the text editor either). The second message, edclose, requires a method that should be added to your object at initialization time. The third message, edSave, allows you to gain access to the text before it is saved, or save it yourself.

#### See also

[Showing a Text Editor](#)

## 38.30 Presets

Max contains a preset object that has the ability to send preset messages to some or all of the objects (clients) in a Patcher window.

Collaboration diagram for Presets:



## Functions

- void [preset\\_store](#) (char \*fmt,...)
 

*Give the preset object a general message to restore the current state of your object.*
- void [preset\\_set](#) ([t\\_object](#) \*obj, [t\\_atom\\_long](#) val)
 

*Restore the state of your object with a set message.*
- void [preset\\_int](#) ([t\\_object](#) \*x, [t\\_atom\\_long](#) n)
 

*Restore the state of your object with an int message.*

### 38.30.1 Detailed Description

Max contains a preset object that has the ability to send preset messages to some or all of the objects (clients) in a Patcher window.

The preset message, sent when the user is storing a preset, is just a request for your object to tell the preset object how to restore your internal state to what it is now. Later, when the user executes a preset, the preset object will send you back the message you had previously said you wanted.

The dialog goes something like this:

- During a store...preset object to Client object(s): hello, this is the preset message—tell me how to restore your state
- Client object to preset object: send me int 34 (for example)
- During an execute...preset object to Client object: int 34

The client object won't know the difference between receiving int 34 from a preset object and receiving a 34 in its leftmost inlet.

It's not mandatory for your object to respond to the preset message, but it is something that will make users happy. All Max user interface objects currently respond to preset messages. Note that if your object is not a user interface object and implements a preset method, the user will need to connect the outlet of the preset object to its leftmost inlet in order for it to be sent a preset message when the user stores a preset.

Here's an example of using [preset\\_store\(\)](#) that specifies that the object would like to receive a set message. We assume it has one field, myvalue, which it would like to save and restore.

```
void myobject_preset(myobject **x)
{
    preset_store("ossl", x, ob_sym(x), gensym("set"), x->myvalue);
}
```

When this preset is executed, the object will receive a set message whose argument will be the value of myvalue. Note that the same thing can be accomplished more easily with [preset\\_set\(\)](#) and [preset\\_int\(\)](#).

Don't pass more than 12 items to [preset\\_store\(\)](#). If you want to store a huge amount of data in a preset, use [binbuf\\_insert\(\)](#).

The following example locates the Binbuf into which the preset data is being collected, then calls [binbuf\\_insert\(\)](#) on a previously prepared array of Atoms. It assumes that the state of your object can be restored with a set message.

```
void myobject_preset(myObject **x)
{
    void *preset_buf; // Binbuf that stores the preset
    short atomCount; // number of atoms you're storing
    t_atom atomArray[SOMESIZE]; // array of atoms to be stored
    // 1. prepare the preset "header" information
    atom_setobj(atomArray, x);
    atom_setsym(atomArray+1, ob_sym(x));
    atom_setsym(atomArray+2, gensym("set"));
    // fill atomArray+3 with object's state here and set atomCount
    // 2. find the Binbuf
    preset_buf = gensym("_preset")->s_thing;
    // 3. store the data
    if (preset_buf) {
        binbuf_insert(preset_buf, NIL, atomCount, atomArray);
    }
}
```

### 38.30.2 Function Documentation

#### 38.30.2.1 preset\_int()

```
void preset_int (
    t_object * x,
    t_atom_long n )
```

Restore the state of your object with an int message.

This function causes an int message with the argument value to be sent to your object from the preset object when the user executes a preset. All of the existing user interface objects use the int message for restoring their state when a preset is executed.

##### Parameters

x	Your object.
n	Current value of your object.

#### 38.30.2.2 preset\_set()

```
void preset_set (
    t_object * obj,
    t_atom_long val )
```

Restore the state of your object with a set message.

This function causes a set message with the argument value to be sent to your object from the preset object when the user executes a preset.

##### Parameters

obj	Your object.
val	Current value of your object.

#### 38.30.2.3 preset\_store()

```
void preset_store (
    char * fmt,
    ... )
```

Give the preset object a general message to restore the current state of your object.

This is a general preset function for use when your object's state cannot be restored with a simple int or set message. The example below shows the expected format for specifying what your current state is to a preset object. The first thing you supply is your object itself, followed by the symbol that is the name of your object's class (which you can retrieve from your object using the macro ob\_sym, declared in ext\_mess.h). Next, supply the symbol that specifies the message you want receive (a method for which had better be defined in your class), followed by the arguments to this message—the current values of your object's fields.

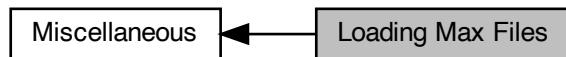
#### Parameters

<i>fmt</i>	C string containing one or more letters corresponding to the types of each element of the message. s for Symbol, l for long, or f for float.
...	Elements of the message used to restore the state of your object, passed directly to the function as Symbols, longs, or floats. See below for an example that conforms to what the preset object expects.

## 38.31 Loading Max Files

Several high-level functions permit you to load patcher files.

Collaboration diagram for Loading Max Files:



### Functions

- short `readtohandle` (C74\_CONST char \*name, short volume, char \*\*\*h, long \*sizep)
   
*Load a data file into a handle.*
- void \* `fileload` (C74\_CONST char \*name, short vol)
   
*Load a patcher file by name and volume reference number.*
- void \* `intload` (C74\_CONST char \*name, short volume, t\_symbol \*s, short ac, t\_atom \*av, short couldedit)
   
*Pass arguments to Max files when you open them.*
- void \* `stringload` (C74\_CONST char \*name)
   
*Load a patcher file located in the Max search path by name.*

### 38.31.1 Detailed Description

Several high-level functions permit you to load patcher files.

These can be used in sophisticated objects that use Patcher objects to perform specific tasks.

### 38.31.2 Function Documentation

#### 38.31.2.1 fileload()

```
void* fileload (
    C74_CONST char * name,
    short vol )
```

Load a patcher file by name and volume reference number.

##### Parameters

<i>name</i>	Filename of the patcher file to load (C string).
<i>vol</i>	Path ID specifying the location of the file.

##### Returns

If the file is found, fileload tries to open the file, evaluate it, open a window, and bring it to the front. A pointer to the newly created Patcher is returned if loading is successful, otherwise, if the file is not found or there is insufficient memory, zero is returned.

#### 38.31.2.2 intload()

```
void* intload (
    C74_CONST char * name,
    short volume,
    t_symbol * s,
    short ac,
    t_atom * av,
    short couldedit )
```

Pass arguments to Max files when you open them.

This function loads the specified file and returns a pointer to the created object. Historically, [intload\(\)](#) was used to open patcher files, whether they are in text or Max binary format. It could also open table files whose contents begin with the word "table".

##### Parameters

<i>name</i>	Name of the file to open.
<i>volume</i>	Path ID specifying the location of the file.
<i>s</i>	A symbol.
<i>ac</i>	Count of t_atoms in av. To properly open a patcher file, ac should be 9.
<i>av</i>	Array of t_atoms that will replace the changeable arguments 1-9. The default behavior could be to set
<b>Cycling '74</b>	all these to t_atoms of type <a href="#">A_LONG</a> with a value of 0.
<i>couldedit</i>	If non-zero and the file is not a patcher file, the file is opened as a text file.

**Returns**

If couldedit is non-zero and the file is not a patcher file, it is made into a text editor, and intload() returns 0. If couldedit is non-zero, [intload\(\)](#) will alert the user to an error and return 0. If there is no error, the value returned will be a pointer to a patcher or table object.

**38.31.2.3 readtohandle()**

```
short readtohandle (
    C74_CONST char * name,
    short volume,
    char *** h,
    long * sizep )
```

Load a data file into a handle.

This is a low-level routine used for reading text and data files. You specify the file's name and Path ID, as well as a pointer to a Handle.

**Parameters**

<i>name</i>	Name of the patcher file to load.
<i>volume</i>	Path ID specifying the location of the file.
<i>h</i>	Pointer to a handle variable that will receive the handle that contains the data in the file.
<i>sizep</i>	Size of the handle returned in <i>h</i> .

**Returns**

If the file is found, readtohandle creates a Handle, reads all the data in the file into it, assigns the handle to the variable *hp*, and returns the size of the data in *size*. readtohandle returns 0 if the file was opened and read successfully, and non-zero if there was an error.

**38.31.2.4 stringload()**

```
void* stringload (
    C74_CONST char * name )
```

Load a patcher file located in the Max search path by name.

This function searches for a patcher file, opens it, evaluates it as a patcher file, opens a window for the patcher and brings it to the front. You need only specify a filename and Max will look through its search path for the file. The search path begins with the current 'default volume' that is often the volume of the last opened patcher file, then the folders specified in the File Preferences dialog, searched depth first, then finally the folder that contains the Max application.

**Parameters**

<code>name</code>	Filename of the patcher file to load (C string).
-------------------	--

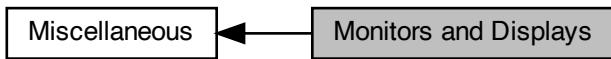
**Returns**

If `stringload()` returns a non-zero result, you can later use `freeobject()` to close the patcher, or just let users do it themselves. If `stringload()` returns zero, no file with the specified name was found or there was insufficient memory to open it.

## 38.32 Monitors and Displays

Functions for finding our information about the environment.

Collaboration diagram for Monitors and Displays:



## Functions

- long `jmonitor_getnumdisplays` (void)
 

*Return the number of monitors on which can be displayed.*
- void `jmonitor_getdisplayrect` (long workarea, long displayindex, `t_rect` \*rect)
 

*Return the `t_rect` for a given display.*
- void `jmonitor_getdisplayrect_foralldisplays` (long workarea, `t_rect` \*rect)
 

*Return a union of all display rects.*
- void `jmonitor_getdisplayrect_forpoint` (long workarea, `t_pt` pt, `t_rect` \*rect)
 

*Return the `t_rect` for the display on which a point exists.*
- double `jmonitor_getdisplayscalefactor` (long displayindex)
 

*Return the scale factor for the display on which a point exists.*
- double `jmonitor_getdisplayscalefactor_forpoint` (`t_pt` pt)
 

*Return the scale factor for the display on which a point exists.*
- `t_pt jmonitor_scale_pt` (`t_pt` unscaled\_pt)
 

*take an unscaled point and convert it to a scaled point note – most APIs take scaled coordinates so drawing doesn't need to consider scale factor for each monitor*
- `t_pt jmonitor_unscale_pt` (`t_pt` scaled\_pt)
 

*take an unscaled point and convert it to a scaled point note – most APIs take scaled coordinates so drawing doesn't need to consider scale factor for each monitor*

### 38.32.1 Detailed Description

Functions for finding our information about the environment.

### 38.32.2 Function Documentation

#### 38.32.2.1 jmonitor\_getdisplayrect()

```
void jmonitor_getdisplayrect (
    long workarea,
    long displayindex,
    t_rect * rect )
```

Return the `t_rect` for a given display.

##### Parameters

<code>workarea</code>	Set workarea non-zero to clip out things like dock / task bar.
<code>displayindex</code>	The index number for a monitor. The primary monitor has an index of 0.
<code>rect</code>	The address of a valid <code>t_rect</code> whose values will be filled-in upon return.

#### 38.32.2.2 jmonitor\_getdisplayrect\_foralldisplays()

```
void jmonitor_getdisplayrect_foralldisplays (
    long workarea,
    t_rect * rect )
```

Return a union of all display rects.

##### Parameters

<code>workarea</code>	Set workarea non-zero to clip out things like dock / task bar.
<code>rect</code>	The address of a valid <code>t_rect</code> whose values will be filled-in upon return.

#### 38.32.2.3 jmonitor\_getdisplayrect\_forpoint()

```
void jmonitor_getdisplayrect_forpoint (
    long workarea,
```

```
t_pt pt,
t_rect * rect )
```

Return the `t_rect` for the display on which a point exists.

#### Parameters

<code>workarea</code>	Set workarea non-zero to clip out things like dock / task bar.
<code>pt</code>	A point, for which the monitor will be determined and the rect returned.
<code>rect</code>	The address of a valid <code>t_rect</code> whose values will be filled-in upon return.

### 38.32.2.4 jmonitor\_getdisplayscalefactor()

```
double jmonitor_getdisplayscalefactor (
    long displayindex )
```

Return the scale factor for the display on which a point exists.

#### Parameters

<code>displayindex</code>	Index of the monitor whose scale factor will be returned.
---------------------------	---

### 38.32.2.5 jmonitor\_getdisplayscalefactor\_forpoint()

```
double jmonitor_getdisplayscalefactor_forpoint (
    t_pt pt )
```

Return the scale factor for the display on which a point exists.

#### Parameters

<code>pt</code>	A point, for which the monitor will be determined and the scale factor returned.
-----------------	--

### 38.32.2.6 jmonitor\_getnumdisplays()

```
long jmonitor_getnumdisplays (
    void )
```

Return the number of monitors on which can be displayed.

**Returns**

The number of monitors.

**38.32.2.7 jmonitor\_scale\_pt()**

```
t_pt jmonitor_scale_pt (
    t_pt unscaled_pt )
```

take an unscaled point and convert it to a scaled point note – most APIs take scaled coordinates so drawing doesn't need to consider scale factor for each monitor

**Parameters**

<i>unscaled</i> <i>_pt</i>	the point to be scaled, should be in pixels on the global "virtual" screen consisting of all monitors
-------------------------------	---

**Returns**

pointer to receive the scaled point, scaled to normalize based on various scale factors and monitor arrangement

**38.32.2.8 jmonitor\_unscale\_pt()**

```
t_pt jmonitor_unscale_pt (
    t_pt scaled_pt )
```

take an unscaled point and convert it to a scaled point note – most APIs take scaled coordinates so drawing doesn't need to consider scale factor for each monitor

**Parameters**

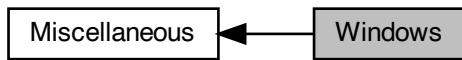
<i>scaled</i> <i>_pt</i>	the point to be unscaled, should be in normalized coordinates, scaled based on scale factors and monitor arrangement
-----------------------------	--

**Returns**

pointer to receive the unscaled point, which will be pixel-based coordinates relative to the main monitor origin

## 38.33 Windows

Collaboration diagram for Windows:



### Functions

- `t_object * jwind_getactive (void)`  
*Get the current window, if any.*
- `long jwind_getcount (void)`  
*Determine how many windows exist.*
- `t_object * jwind_getat (long index)`  
*Return a pointer to the window with a given index.*

#### 38.33.1 Detailed Description

#### 38.33.2 Function Documentation

##### 38.33.2.1 `jwind_getactive()`

```
t_object* jwind_getactive (
    void )
```

Get the current window, if any.

**Returns**

A pointer to the current window, if there is one. Otherwise returns NULL.

##### 38.33.2.2 `jwind_getat()`

```
t_object* jwind_getat (
    long index )
```

Return a pointer to the window with a given index.

**Parameters**

<i>index</i>	Get window at index (0 to count-1).
--------------	-------------------------------------

**Returns**

A pointer to a window object.

**38.33.2.3 jwind\_getcount()**

```
long jwind_getcount (
    void )
```

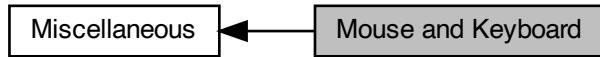
Determine how many windows exist.

**Returns**

The number of windows.

**38.34 Mouse and Keyboard**

Collaboration diagram for Mouse and Keyboard:

**Enumerations**

- enum `t_jmouse_cursortype` {
 `JMOUSE_CURSOR_NONE`, `JMOUSE_CURSOR_ARROW`, `JMOUSE_CURSOR_WAIT`, `JMOUSE_CURSOR_IBeam`,
 ,
 `JMOUSE_CURSOR_CROSSHAIR`, `JMOUSE_CURSOR_COPYING`, `JMOUSE_CURSOR_POINTINGHAND`,
 `JMOUSE_CURSOR_DRAGGINGHAND`,
 `JMOUSE_CURSOR_RESIZE_LEFTRIGHT`, `JMOUSE_CURSOR_RESIZE_UPDOWN`, `JMOUSE_CURSOR_RESIZE_FOURWAY`,
 , `JMOUSE_CURSOR_RESIZE_TOPEDGE`,
 `JMOUSE_CURSOR_RESIZE_BOTTOMEDGE`, `JMOUSE_CURSOR_RESIZE_LEFTEDGE`, `JMOUSE_CURSOR_RESIZE_RIGHTEDGE`,
 , `JMOUSE_CURSOR_RESIZE_TOPLEFTCORNER`,
 `JMOUSE_CURSOR_RESIZE_TOPRIGHTCORNER`, `JMOUSE_CURSOR_RESIZE_BOTTOMLEFTCORNER`,
 `JMOUSE_CURSOR_RESIZE_BOTTOMRIGHTCORNER` }

*Mouse cursor types.*

- enum `t_modifiers` {
 `eCommandKey` , `eShiftKey` , `eControlKey` , `eAltKey` ,
 `eLeftButton` , `eRightButton` , `eMiddleButton` , `ePopupMenu` ,
 `eCapsLock` , `eAutoRepeat` }

*Bit mask values for various meta-key presses on the keyboard.*

## Functions

- `t_modifiers jkeyboard_getcurrentmodifiers (void)`

*Return the last known combination of modifier keys being held by the user.*

- `t_modifiers jkeyboard_getcurrentmodifiers_realtime (void)`

*Return the current combination of modifier keys being held by the user.*

- void `jmouse_getposition_global (int *x, int *y)`

*Get the position of the mouse cursor in screen coordinates.*

- void `jmouse_setposition_global (int x, int y)`

*Set the position of the mouse cursor in screen coordinates.*

- void `jmouse_setposition_view (t_object *patcherview, double cx, double cy)`

*Set the position of the mouse cursor relative to the patcher canvas coordinates.*

- void `jmouse_setposition_box (t_object *patcherview, t_object *box, double bx, double by)`

*Set the position of the mouse cursor relative to a box within the patcher canvas coordinates.*

- void `jmouse_setcursor (t_object *patcherview, t_object *box, t_jmouse_cursortype type)`

*Set the mouse cursor.*

### 38.34.1 Detailed Description

### 38.34.2 Enumeration Type Documentation

#### 38.34.2.1 `t_jmouse_cursortype`

enum `t_jmouse_cursortype`

Mouse cursor types.

Enumerator

<code>JMOUSE_CURSOR_NONE</code>	None.
<code>JMOUSE_CURSOR_ARROW</code>	Arrow.
<code>JMOUSE_CURSOR_WAIT</code>	Wait.
<code>JMOUSE_CURSOR_IBEAM</code>	I-Beam.
<code>JMOUSE_CURSOR_CROSSHAIR</code>	Crosshair.
<code>JMOUSE_CURSOR_COPYING</code>	Copying.
<code>JMOUSE_CURSOR_POINTINGHAND</code>	Pointing Hand.

**Enumerator**

JMOUSE_CURSOR_DRAGGINGHAND	Dragging Hand.
JMOUSE_CURSOR_RESIZE_LEFTRIGHT	Left-Right.
JMOUSE_CURSOR_RESIZE_UPDOWN	Up-Down.
JMOUSE_CURSOR_RESIZE_FOURWAY	Four Way.
JMOUSE_CURSOR_RESIZE_TOPEDGE	Top Edge.
JMOUSE_CURSOR_RESIZE_BOTTOMEDGE	Bottom Edge.
JMOUSE_CURSOR_RESIZE_LEFTEDGE	Left Edge.
JMOUSE_CURSOR_RESIZE_RIGHTEDGE	Right Edge.
JMOUSE_CURSOR_RESIZE_TOPLEFTCORNER	Top-Left Corner.
JMOUSE_CURSOR_RESIZE_TOPRIGHTCORNER	Top-Right Corner.
JMOUSE_CURSOR_RESIZE_BOTTOMLEFTCORNER	Bottom-Left Corner.
JMOUSE_CURSOR_RESIZE_BOTTOMRIGHTCORNER	Bottom-Right Corner.

**38.34.2.2 t\_modifiers**

```
enum t_modifiers
```

Bit mask values for various meta-key presses on the keyboard.

**Enumerator**

eCommandKey	Command Key.
eShiftKey	Shift Key.
eControlKey	Control Key.
eAltKey	Alt Key.
eLeftButton	Left mouse button.
eRightButton	Right mouse button.
eMiddleButton	Middle mouse button.
ePopupMenu	Popup Menu (contextual menu requested)
eCapsLock	Caps lock.
eAutoRepeat	Key is generated by key press auto-repeat.

**38.34.3 Function Documentation****38.34.3.1 jkeyboard\_getcurrentmodifiers()**

```
t_modifiers jkeyboard_getcurrentmodifiers (
    void   )
```

Return the last known combination of modifier keys being held by the user.

**Returns**

The current modifier keys that are activated.

### 38.34.3.2 jkeyboard\_getcurrentmodifiers\_realtime()

```
t_modifiers jkeyboard_getcurrentmodifiers_realtime (
    void )
```

Return the current combination of modifier keys being held by the user.

**Returns**

The current modifier keys that are activated.

### 38.34.3.3 jmouse\_getposition\_global()

```
void jmouse_getposition_global (
    int * x,
    int * y )
```

Get the position of the mouse cursor in screen coordinates.

**Parameters**

x	The address of a variable to hold the x-coordinate upon return.
y	The address of a variable to hold the y-coordinate upon return.

### 38.34.3.4 jmouse\_setcursor()

```
void jmouse_setcursor (
    t_object * patcherview,
    t_object * box,
    t_jmouse_cursortype type )
```

Set the mouse cursor.

**Parameters**

<i>patcherview</i>	The patcherview for which the cursor should be applied.
<i>box</i>	The box for which the cursor should be applied.
<i>type</i>	The type of cursor for the mouse to use.

**38.34.3.5 jmouse\_setposition\_box()**

```
void jmouse_setposition_box (
    t_object * patcherview,
    t_object * box,
    double bx,
    double by )
```

Set the position of the mouse cursor relative to a box within the patcher canvas coordinates.

**Parameters**

<i>patcherview</i>	The patcherview containing the box upon which the mouse coordinates are based.
<i>box</i>	The box upon which the mouse coordinates are based.
<i>bx</i>	The new x-coordinate of the mouse cursor position.
<i>by</i>	The new y-coordinate of the mouse cursor position.

**38.34.3.6 jmouse\_setposition\_global()**

```
void jmouse_setposition_global (
    int x,
    int y )
```

Set the position of the mouse cursor in screen coordinates.

**Parameters**

<i>x</i>	The new x-coordinate of the mouse cursor position.
<i>y</i>	The new y-coordinate of the mouse cursor position.

**38.34.3.7 jmouse\_setposition\_view()**

```
void jmouse_setposition_view (
```

```
t_object * patcherview,
double cx,
double cy )
```

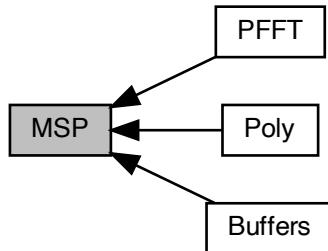
Set the position of the mouse cursor relative to the patcher canvas coordinates.

#### Parameters

<i>patcherview</i>	The patcherview upon which the mouse coordinates are based.
<i>cx</i>	The new x-coordinate of the mouse cursor position.
<i>cy</i>	The new y-coordinate of the mouse cursor position.

## 38.35 MSP

Collaboration diagram for MSP:



## Modules

- **Buffers**

*Your object can access shared data stored in an MSP buffer~ object.*

- **PFFT**

*When an object is instantiated, it is possible to determine if it is being created in pfft~ context in the new method.*

- **Poly**

*If your object is instantiated as a voice of a poly~ object, it is possible both to determine this context and to determine information about the specific voice.*

## Data Structures

- struct `t_pxdata`  
*Common struct for MSP objects.*
- struct `t_pxobject`  
*Header for any non-ui signal processing object.*
- struct `t_signal`  
*The signal data structure.*
- struct `t_pxjbox`  
*Header for any ui signal processing object.*

## Macros

- `#define Z_NO_INPLACE`  
*flag indicating the object doesn't want signals in place*
- `#define Z_PUT_LAST`  
*when list of ugens is resorted, put this object at end*
- `#define Z_PUT_FIRST`  
*when list of ugens is resorted, put this object at beginning*
- `#define PI`  
*The pi constant.*
- `#define TWOPI`  
*Twice the pi constant.*
- `#define PIOVERTWO`  
*Half of the pi constant.*
- `#define dsp_setup`  
*This is commonly used rather than directly calling `z_dsp_setup()` in MSP objects.*
- `#define dsp_free`  
*This is commonly used rather than directly calling `z_dsp_free()` in MSP objects.*

## Typedefs

- `typedef void * t_vptr`  
*A void pointer.*
- `typedef void * vptr`  
*A void pointer.*
- `typedef float t_float`  
*A float – always a 32 bit floating point number.*
- `typedef double t_double`  
*A double – always a 64 bit floating point number.*
- `typedef double t_sample`  
*A sample value – width determined by MSP version.*

## Enumerations

- enum { `SYS_MAXBLKSIZE` }
- MSP System Properties.*

## Functions

- int `sys_getmaxblksize` (void)  
*Query MSP for the maximum global vector (block) size.*
- int `sys_getblksize` (void)  
*Query MSP for the current global vector (block) size.*
- float `sys_getsr` (void)  
*Query MSP for the global sample rate.*
- int `sys_getdspstate` (void)  
*Query MSP to determine whether or not it is running.*
- int `sys_getdspobjdspstate` (`t_object` \*o)  
*Query MSP to determine whether or not a given audio object is in a running dsp chain.*
- void `dsp_add` (`t_perfroutine` f, int n,...)  
*Call this function in your MSP object's dsp method.*
- void `dsp_advv` (`t_perfroutine` f, int n, void \*\*vector)  
*Call this function in your MSP object's dsp method.*
- void `z_dsp_setup` (`t_pxobject` \*x, long nsignals)  
*Call this routine after creating your object in the new instance routine with `object_alloc()`.*
- void `z_dsp_free` (`t_pxobject` \*x)  
*This function disposes of any memory used by proxies allocated by `dsp_setup()`.*
- void `class_dspinit` (`t_class` \*c)  
*This routine must be called in your object's initialization routine.*
- void `class_dspinitbbox` (`t_class` \*c)  
*This routine must be called in your object's initialization routine.*

## Variables

- `BEGIN_USING_C_LINKAGE` typedef `t_int` \*(\*`t_perfroutine`)(`t_int` \*args)  
*A function pointer for the audio perform routine used by MSP objects to process blocks of samples.*

### 38.35.1 Detailed Description

### 38.35.2 Macro Definition Documentation

#### 38.35.2.1 PI

```
#define PI
```

The pi constant.

### 38.35.2.2 PIOVERTWO

```
#define PIOVERTWO
```

Half of the pi constant.

### 38.35.2.3 TWOPI

```
#define TWOPI
```

Twice the pi constant.

## 38.35.3 Typedef Documentation

### 38.35.3.1 t\_double

```
typedef double t_double
```

A double – always a 64 bit floating point number.

### 38.35.3.2 t\_float

```
typedef float t_float
```

A float – always a 32 bit floating point number.

### 38.35.3.3 t\_sample

```
typedef double t_sample
```

A sample value – width determined by MSP version.

### 38.35.3.4 `t_vptr`

```
typedef void* t_vptr
```

A void pointer.

### 38.35.3.5 `vptr`

```
typedef void* vptr
```

A void pointer.

## 38.35.4 Enumeration Type Documentation

### 38.35.4.1 `anonymous enum`

```
anonymous enum
```

MSP System Properties.

Enumerator

SYS_MAXBLKSIZE	a good number for a maximum signal vector size
----------------	--

## 38.35.5 Function Documentation

### 38.35.5.1 `class_dspinit()`

```
void class_dspinit (
    t_class * c )
```

This routine must be called in your object's initialization routine.

It adds a set of methods to your object's class that are called by MSP to build the DSP call chain. These methods function entirely transparently to your object so you don't have to worry about them. However, you should avoid binding anything to their names: signal, userconnect, nsiginlets, and enable.

This routine is for non-user-interface objects only (where the first item in your object's struct is a `t_pxobject`). It must be called prior to calling `class_register()` for your class.

**Parameters**

<code>c</code>	The class to make dsp-ready.
----------------	------------------------------

**See also**

[class\\_dspinitjbox\(\)](#)

**38.35.5.2 class\_dspinitjbox()**

```
void class_dspinitjbox (
    t_class * c )
```

This routine must be called in your object's initialization routine.

It adds a set of methods to your object's class that are called by MSP to build the DSP call chain. These methods function entirely transparently to your object so you don't have to worry about them. However, you should avoid binding anything to their names: signal, userconnect, nsiginlets, and enable.

This routine is for user-interface objects only (where the first item in your object's struct is a [t\\_jbox](#)).

**Parameters**

<code>c</code>	The class to make dsp-ready.
----------------	------------------------------

**See also**

[class\\_dspinit\(\)](#)

**38.35.5.3 dsp\_add()**

```
void dsp_add (
    t_perfroutine f,
    int n,
    ... )
```

Call this function in your MSP object's dsp method.

This function adds your object's perform method to the DSP call chain and specifies the arguments it will be passed. n, the number of arguments to your perform method, should be followed by n additional arguments, all of which must be the size of a pointer or a long.

### Parameters

<i>f</i>	The perform routine to use for processing audio.
<i>n</i>	The number of arguments that will follow
...	The arguments that will be passed to the perform routine.

### See also

[The DSP Method and Perform Routine](#)

[Using Connection Information](#)

### 38.35.5.4 `dsp_addv()`

```
void dsp_addv (
    t_perfroutine f,
    int n,
    void ** vector )
```

Call this function in your MSP object's `dsp` method.

Use `dsp_addv()` to add your object's perform routine to the DSP call chain and specify its arguments in an array rather than as arguments to a function.

### Parameters

<i>f</i>	The perform routine to use for processing audio.
<i>n</i>	The number of arguments that will follow in the <code>vector</code> parameter.
<i>vector</i>	The arguments that will be passed to the perform routine.

### See also

[The DSP Method and Perform Routine](#)

[Using Connection Information](#)

### 38.35.5.5 `sys_getblksize()`

```
int sys_getblksize (
    void )
```

Query MSP for the current global vector (block) size.

### Returns

The current global vector size for the MSP environment.

### 38.35.5.6 sys\_getdspobjdspstate()

```
int sys_getdspobjdspstate (
    t_object * o )
```

Query MSP to determine whether or not a given audio object is in a running dsp chain.

This is preferable over [sys\\_getdspstate\(\)](#) since global audio can be on but an object could be in a patcher that is not running.

#### Returns

Returns true if the MSP object is in a patcher that has audio on, otherwise returns false.

### 38.35.5.7 sys\_getdspstate()

```
int sys_getdspstate (
    void )
```

Query MSP to determine whether or not it is running.

#### Returns

Returns true if the DSP is turned on, otherwise returns false.

### 38.35.5.8 sys\_getmaxblksize()

```
int sys_getmaxblksize (
    void )
```

Query MSP for the maximum global vector (block) size.

#### Returns

The maximum global vector size for the MSP environment.

### 38.35.5.9 sys\_getsr()

```
float sys_getsr (
    void )
```

Query MSP for the global sample rate.

#### Returns

The global sample rate of the MSP environment.

### 38.35.5.10 z\_dsp\_free()

```
void z_dsp_free (
    t_pxobject * x )
```

This function disposes of any memory used by proxies allocated by [dsp\\_setup\(\)](#).

It also notifies the signal compiler that the DSP call chain needs to be rebuilt if signal processing is active. You should be sure to call this before de-allocating any memory that might be in use by your object's perform routine, in the event that signal processing is on when your object is freed.

#### Parameters

x	The object to free.
---	---------------------

#### See also

[dsp\\_free](#)

### 38.35.5.11 z\_dsp\_setup()

```
void z_dsp_setup (
    t_pxobject * x,
    long nsignals )
```

Call this routine after creating your object in the new instance routine with [object\\_alloc\(\)](#).

Cast your object to [t\\_pxobject](#) as the first argument, then specify the number of signal inputs your object will have. [dsp\\_setup\(\)](#) initializes fields of the [t\\_pxobject](#) header and allocates any proxies needed (if num\_signal\_inputs is greater than 1).

Some signal objects have no inputs; you should pass 0 for num\_signal\_inputs in this case. After calling [dsp\\_setup\(\)](#), you can create additional non-signal inlets using [intin\(\)](#), [floatin\(\)](#), or [inlet\\_new\(\)](#).

**Parameters**

<code>x</code>	Your object's pointer.
<code>nsignals</code>	The number of signal/proxy inlets to create for the object.

**See also**

[dsp\\_setup](#)

## 38.35.6 Variable Documentation

### 38.35.6.1 `t_perfroutine`

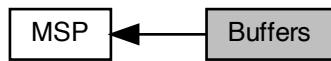
```
BEGIN_USING_C_LINKAGE typedef t_int*(* t_perfroutine) (t_int *args) (
    t_int * args )
```

A function pointer for the audio perform routine used by MSP objects to process blocks of samples.

## 38.36 Buffers

Your object can access shared data stored in an MSP buffer~ object.

Collaboration diagram for Buffers:



## Data Structures

- struct `t_buffer_info`  
*Common buffer~ data/metadata.*

## Typedefs

- `typedef struct _buffer_ref t_buffer_ref`  
*A buffer~ reference.*
- `typedef t_object t_buffer_obj`  
*A buffer~ object.*

## Functions

- `BEGIN_USING_C_LINKAGE t_buffer_ref * buffer_ref_new (t_object *self, t_symbol *name)`  
*Create a reference to a buffer~ object by name.*
- `void buffer_ref_set (t_buffer_ref *x, t_symbol *name)`  
*Change a buffer reference to refer to a different buffer~ object by name.*
- `t_atom_long buffer_ref_exists (t_buffer_ref *x)`  
*Query to find out if a buffer~ with the referenced name actually exists.*
- `t_buffer_obj * buffer_ref_getobject (t_buffer_ref *x)`  
*Query a buffer reference to get the actual buffer~ object being referenced, if it exists.*
- `t_max_err buffer_ref_notify (t_buffer_ref *x, t_symbol *s, t_symbol *msg, void *sender, void *data)`  
*Your object needs to handle notifications issued by the buffer~ you reference.*
- `void buffer_view (t_buffer_obj *buffer_object)`  
*Open a viewer window to display the contents of the buffer~.*
- `float * buffer_locksamples (t_buffer_obj *buffer_object)`  
*Claim the buffer~ and get a pointer to the first sample in memory.*
- `void buffer_unlocksamples (t_buffer_obj *buffer_object)`  
*Release your claim on the buffer~ contents so that other objects may read/write to the buffer~.*
- `t_atom_long buffer_getchannelcount (t_buffer_obj *buffer_object)`  
*Query a buffer~ to find out how many channels are present in the buffer content.*
- `t_atom_long buffer_getframecount (t_buffer_obj *buffer_object)`  
*Query a buffer~ to find out how many frames long the buffer content is in samples.*
- `t_atom_float buffer_getsamplerate (t_buffer_obj *buffer_object)`  
*Query a buffer~ to find out its native sample rate in samples per second.*
- `t_atom_float buffer_getmillisamplerate (t_buffer_obj *buffer_object)`  
*Query a buffer~ to find out its native sample rate in samples per millisecond.*
- `t_max_err buffer_setpadding (t_buffer_obj *buffer_object, t_atom_long samplecount)`  
*Set the number of samples with which to zero-pad the buffer~'s contents.*
- `t_max_err buffer_setdirty (t_buffer_obj *buffer_object)`  
*Set the buffer's dirty flag, indicating that changes have been made.*
- `t_symbol * buffer_getfilename (t_buffer_obj *buffer_object)`  
*Retrieve the name of the last file to be read by a buffer~.*

### 38.36.1 Detailed Description

Your object can access shared data stored in an MSP buffer~ object.

Similar to table and coll objects, buffer~ objects are bound to a [t\\_symbol](#) from which you can direct gain access to the t\_buffer struct. This is potentially dangerous, and not guaranteed to be forward (or backward) compatible. Beginning with Max 6.1, developers accessing buffer~ objects are encouraged to use the [t\\_buffer\\_ref](#) API. The [t\\_buffer\\_ref](#) API provides many enhancements to improve thread-safety, simplify your perform routine, and manage the binding to the buffer~ object.

A class that accesses a buffer~ is the simpwave~ object included with Max SDK example projects.

While the Max 6 signal processing chain operates on 64-bit double-precision floats, the [t\\_buffer\\_obj](#) storage remains as 32-bit single-precision float format. This is essential to maintain backward compatibility with older third-party externals.

If you have written to the buffer~ and thus changed the values of its samples, you should now mark the buffer~ as dirty. This will ensure that objects such as waveform~ update their rendering of the contents of this buffer~. This can be accomplished with the following call:

```
object_method(b, gensym("dirty"));
```

### 38.36.2 Typedef Documentation

#### 38.36.2.1 [t\\_buffer\\_obj](#)

```
typedef t_object t_buffer_obj
```

A buffer~ object.

This represents the actual buffer~ object. You can use this to send messages, query attributes, etc. of the actual buffer object referenced by a [t\\_buffer\\_ref](#).

#### 38.36.2.2 [t\\_buffer\\_ref](#)

```
typedef struct _buffer_ref t_buffer_ref
```

A buffer~ reference.

Use this struct to represent a reference to a buffer~ object in Max. Use the `buffer_ref_getbuffer()` call to return a pointer to the buffer. You can then make calls on the buffer itself.

### 38.36.3 Function Documentation

#### 38.36.3.1 [buffer\\_getchannelcount\(\)](#)

```
t_atom_long buffer_getchannelcount (
    t_buffer_obj * buffer_object )
```

Query a buffer~ to find out how many channels are present in the buffer content.

**Parameters**

<i>buffer_object</i>	the buffer object
----------------------	-------------------

**Returns**

the number of channels in the buffer

**38.36.3.2 buffer\_getfilename()**

```
t_symbol* buffer_getfilename (
    t_buffer_obj * buffer_object )
```

Retrieve the name of the last file to be read by a buffer~.

(Not the last file written).

**Parameters**

<i>buffer_object</i>	the buffer object
----------------------	-------------------

**Returns**

The name of the file last read, or gensym("") if no files have been read.

**Version**

Introduced in Max 7.0.1

**38.36.3.3 buffer\_getframecount()**

```
t_atom_long buffer_getframecount (
    t_buffer_obj * buffer_object )
```

Query a buffer~ to find out how many frames long the buffer content is in samples.

**Parameters**

<i>buffer_object</i>	the buffer object
----------------------	-------------------

**Returns**

the number of frames in the buffer

**38.36.3.4 buffer\_getmillisamplerate()**

```
t_atom_float buffer_getmillisamplerate (
    t_buffer_obj * buffer_object )
```

Query a buffer~ to find out its native sample rate in samples per millisecond.

**Parameters**

<i>buffer_object</i>	the buffer object
----------------------	-------------------

**Returns**

the sample rate in samples per millisecond

**38.36.3.5 buffer\_getsamplerate()**

```
t_atom_float buffer_getsamplerate (
    t_buffer_obj * buffer_object )
```

Query a buffer~ to find out its native sample rate in samples per second.

**Parameters**

<i>buffer_object</i>	the buffer object
----------------------	-------------------

**Returns**

the sample rate in samples per second

**38.36.3.6 buffer\_locksamples()**

```
float* buffer_locksamples (
    t_buffer_obj * buffer_object )
```

Claim the buffer~ and get a pointer to the first sample in memory.

When you are done reading/writing to the buffer you must call [buffer\\_unlocksamples\(\)](#). If the attempt to claim the buffer~ fails the returned pointer will be NULL.

**Parameters**

<i>buffer_object</i>	the buffer object
----------------------	-------------------

**Returns**

a pointer to the first sample in memory, or NULL if the buffer doesn't exist.

**38.36.3.7 buffer\_ref\_exists()**

```
t_atom_long buffer_ref_exists (
    t_buffer_ref * x )
```

Query to find out if a buffer~ with the referenced name actually exists.

**Parameters**

<i>x</i>	the buffer reference
----------	----------------------

**Returns**

non-zero if the buffer~ exists, otherwise zero

**38.36.3.8 buffer\_ref\_getobject()**

```
t_buffer_obj* buffer_ref_getobject (
    t_buffer_ref * x )
```

Query a buffer reference to get the actual buffer~ object being referenced, if it exists.

**Parameters**

<i>x</i>	the buffer reference
----------	----------------------

**Returns**

the buffer object if exists, otherwise NULL

### 38.36.3.9 buffer\_ref\_new()

```
BEGIN_USING_C_LINKAGE t_buffer_ref* buffer_ref_new (
    t_object * self,
    t_symbol * name )
```

Create a reference to a buffer~ object by name.

You must release the buffer reference using [object\\_free\(\)](#) when you are finished using it.

#### Parameters

<i>self</i>	pointer to your object
<i>name</i>	the name of the buffer~

#### Returns

a pointer to your new buffer reference

### 38.36.3.10 buffer\_ref\_notify()

```
t_max_err buffer_ref_notify (
    t_buffer_ref * x,
    t_symbol * s,
    t_symbol * msg,
    void * sender,
    void * data )
```

Your object needs to handle notifications issued by the buffer~ you reference.

You do this by defining a "notify" method. Your notify method should then call this notify method for the [t\\_buffer\\_ref](#).

#### Parameters

<i>x</i>	the buffer reference
<i>s</i>	the registered name of the sending object
<i>msg</i>	then name of the notification/message sent
<i>sender</i>	the pointer to the sending object
<i>data</i>	optional argument sent with the notification/message

#### Returns

a max error code

### 38.36.3.11 buffer\_ref\_set()

```
void buffer_ref_set (
    t_buffer_ref * x,
    t_symbol * name )
```

Change a buffer reference to refer to a different buffer~ object by name.

#### Parameters

x	the buffer reference
name	the name of a different buffer~ to reference

### 38.36.3.12 buffer\_setdirty()

```
t_max_err buffer_setdirty (
    t_buffer_obj * buffer_object )
```

Set the buffer's dirty flag, indicating that changes have been made.

#### Parameters

buffer_object	the buffer object
---------------	-------------------

#### Returns

an error code

### 38.36.3.13 buffer\_setpadding()

```
t_max_err buffer_setpadding (
    t_buffer_obj * buffer_object,
    t_atom_long samplecount )
```

Set the number of samples with which to zero-pad the buffer~'s contents.

The typical application for this need is to pad a buffer with enough room to allow for the reach of a FIR kernel in convolution.

#### Parameters

buffer_object	the buffer object
samplecount	the number of sample to pad the buffer with on each side of the contents

**Returns**

an error code

**38.36.3.14 buffer\_unlocksamples()**

```
void buffer_unlocksamples (
    t_buffer_obj * buffer_object )
```

Release your claim on the buffer~ contents so that other objects may read/write to the buffer~.

**Parameters**

<i>buffer_object</i>	the buffer object
----------------------	-------------------

**38.36.3.15 buffer\_view()**

```
void buffer_view (
    t_buffer_obj * buffer_object )
```

Open a viewer window to display the contents of the buffer~.

**Parameters**

<i>buffer_object</i>	the buffer object
----------------------	-------------------

**38.37 PFFT**

When an object is instantiated, it is possible to determine if it is being created in pfft~ context in the new method.

Collaboration diagram for PFFT:



## Data Structures

- struct `t_pfftpub`

*Public FFT Patcher struct.*

### 38.37.1 Detailed Description

When an object is instantiated, it is possible to determine if it is being created in pfft~ context in the new method.

In the new method (and only at this time), you can check the `s_thing` member of the `t_symbol` '`__pfft~__`'. If this is non-null, then you will have a pointer to a `t_pfftpub` struct.

```
t_pfftpub *pfft_parent = (t_pfftpub*) gensym("__pfft~__")->s_thing;
if (pfft_parent) {
    // in a pfft~ context
}
else {
    // not in a pfft~
}
```

## 38.38 Poly

If your object is instantiated as a voice of a poly~ object, it is possible both to determine this context and to determine information about the specific voice.

Collaboration diagram for Poly:



If your object is instantiated as a voice of a poly~ object, it is possible both to determine this context and to determine information about the specific voice.

This is done by querying the patcher in which your object exists for an associated object, and then calling methods on that object.

```
t_object *patcher = NULL;
t_max_err err = MAX_ERR_NONE;
t_object *assoc = NULL;
method m = NULL;
long voices = -1;
long index = -1;
err = object_obex_lookup(x, gensym("#P"), &patcher);
if (err == MAX_ERR_NONE) {
    object_method(patcher, gensym("getassoc"), &assoc);
    if (assoc) {
        post("found %s", object_classname(assoc)->s_name);
        voices = object_attr_getlong(assoc, gensym("voices"));
        post("total amount of voices: %ld", voices);
        if(m = zgetfn(assoc, gensym("getindex")))
            index = (long)(*m)(assoc, patcher);
        post("index: %ld", index);
    }
}
```

## 38.39 Objects

### Data Structures

- struct `t_messlist`  
*A list of symbols and their corresponding methods, complete with typechecking information.*
- struct `t_tinyobject`  
*The tiny object structure sits at the head of any object to which you may pass messages (and which you may feed to `freeobject()`).*
- struct `t_object`  
*The structure for the head of any object which wants to have inlets or outlets, or support attributes.*

### Macros

- `#define NOGOOD(x)`  
*Returns true if a pointer is not a valid object.*
- `#define object_method_direct(rt, sig, x, s, ...)`  
*do a strongly typed direct call to a method of an object*

### Enumerations

- enum  
*Magic number used to determine if memory pointed to by a `t_object*` is valid.*
- enum  
*Maximum number of arguments that can be passed as a typed-list rather than using `A_GIMME`.*

### Functions

- long `object_classname_compare (void *x, t_symbol *name)`  
*Determines if a particular object is an instance of a given class.*
- void \* `object_alloc (t_class *c)`  
*Allocates the memory for an instance of an object class and initialize its object header.*
- void \* `object_new (t_symbol *name_space, t_symbol *classname,...)`  
*Allocates the memory for an instance of an object class and initialize its object header internal to Max.*
- void \* `object_new_typed (t_symbol *name_space, t_symbol *classname, long ac, t_atom *av)`  
*Allocates the memory for an instance of an object class and initialize its object header internal to Max.*
- `t_max_err object_free (void *x)`  
*Call the free function and release the memory for an instance of an internal object class previously instantiated using `object_new()`, `object_new_typed()` or other new-style object constructor functions (e.g.*
- void \* `object_method (void *x, t_symbol *s,...)`  
*Sends an untyped message to an object.*
- `t_max_err object_method_typed (void *x, t_symbol *s, long ac, t_atom *av, t_atom *rv)`  
*Sends a type-checked message to an object.*
- `t_max_err object_method_typedfun (void *x, t_messlist *mp, t_symbol *s, long ac, t_atom *av, t_atom *rv)`  
*Currently undocumented.*

- method `object_getmethod` (void \*x, `t_symbol` \*s)
 

*Retrieves an object's #method for a particular message selector.*
- `t_symbol * object_classname` (void \*x)
 

*Retrieves an object instance's class name.*
- void \* `object_register` (`t_symbol` \*name\_space, `t_symbol` \*s, void \*x)
 

*Registers an object in a namespace.*
- void \* `object_findregistered` (`t_symbol` \*name\_space, `t_symbol` \*s)
 

*Determines a registered object's pointer, given its namespace and name.*
- `t_max_err object_findregisteredbyptr` (`t_symbol` \*\*name\_space, `t_symbol` \*\*s, void \*x)
 

*Determines the namespace and/or name of a registered object, given the object's pointer.*
- `t_max_err object_getnames` (`t_symbol` \*name\_space, long \*namecount, `t_symbol` \*\*\*names)
 

*Returns all registered names in a namespace.*
- void \* `object_attach` (`t_symbol` \*name\_space, `t_symbol` \*s, void \*x)
 

*Attaches a client to a registered object.*
- `t_max_err object_detach` (`t_symbol` \*name\_space, `t_symbol` \*s, void \*x)
 

*Detach a client from a registered object.*
- `t_max_err object_attach_byptr` (void \*x, void \*registeredobject)
 

*Attaches a client to a registered object.*
- `t_max_err object_attach_byptr_register` (void \*x, void \*object\_to\_attach, `t_symbol` \*reg\_name\_space)
 

*A convenience function wrapping `object_register()` and `object_attach_byptr()`.*
- `t_max_err object_detach_byptr` (void \*x, void \*registeredobject)
 

*Detach a client from a registered object.*
- void \* `object_subscribe` (`t_symbol` \*name\_space, `t_symbol` \*s, `t_symbol` \*classname, void \*x)
 

*Subscribes a client to wait for an object to register.*
- `t_max_err object_unsubscribe` (`t_symbol` \*name\_space, `t_symbol` \*s, `t_symbol` \*classname, void \*x)
 

*Unsubscribe a client from a registered object, detaching if the object is registered.*
- `t_max_err object_unregister` (void \*x)
 

*Removes a registered object from a namespace.*
- `t_max_err object_notify` (void \*x, `t_symbol` \*s, void \*data)
 

*Broadcast a message (with an optional argument) from a registered object to any attached client objects.*
- `t_class * object_class` (void \*x)
 

*Determines the class of a given object.*
- `t_max_err object_getvalueof` (void \*x, long \*ac, `t_atom` \*\*av)
 

*Retrieves the value of an object which supports the `getvalueof/setvalueof` interface.*
- `t_max_err object_setvalueof` (void \*x, long ac, `t_atom` \*av)
 

*Sets the value of an object which supports the `getvalueof/setvalueof` interface.*
- `t_max_err object_obex_lookup` (void \*x, `t_symbol` \*key, `t_object` \*\*val)
 

*Retrieves the value of a data stored in the obex.*
- `t_max_err object_obex_store` (void \*x, `t_symbol` \*key, `t_object` \*val)
 

*Stores data in the object's obex.*
- void `object_obex_dumpout` (void \*x, `t_symbol` \*s, long argc, `t_atom` \*argv)
 

*Sends data from the object's dumpout outlet.*
- `t_dictionary * object_dictionaryarg` (long ac, `t_atom` \*av)
 

*Retrieve a pointer to a dictionary passed in as an atom argument.*
- void \* `object_super_method` (`t_object` \*x, `t_symbol` \*s,...)
 

*Sends an untyped message to an object using superclass methods.*
- void \* `object_this_method` (`t_object` \*x, `t_symbol` \*s,...)

- Sends an untyped message to an object, respects a thread specific class stack from `object_super_method()` calls.*
- `t_max_err object_attr_touch (t_object *x, t_symbol *attrname)`

*Mark an attribute as being touched by some code not from the attribute setter.*
  - `t_max_err object_attr_touch_parse (t_object *x, char *attrnames)`

*Mark one or more attributes as being touched by some code not from the attribute setter.*
  - `t_max_err object_method_parse (t_object *x, t_symbol *s, C74_CONST char *parsestr, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that uses `atom_setparse()` to define the arguments.*
  - `t_max_err object_method_format (t_object *x, t_symbol *s, t_atom *rv, C74_CONST char *fmt,...)`

*Convenience wrapper for `object_method_typed()` that uses `atom_setformat()` to define the arguments.*
  - `t_max_err object_method_char (t_object *x, t_symbol *s, unsigned char v, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that passes a single char as an argument.*
  - `t_max_err object_method_long (t_object *x, t_symbol *s, long v, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that passes a single long integer as an argument.*
  - `t_max_err object_method_float (t_object *x, t_symbol *s, float v, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that passes a single 32bit float as an argument.*
  - `t_max_err object_method_double (t_object *x, t_symbol *s, double v, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that passes a single 64bit float as an argument.*
  - `t_max_err object_method_sym (t_object *x, t_symbol *s, t_symbol *v, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that passes a single `t_symbol*` as an argument.*
  - `t_max_err object_method_obj (t_object *x, t_symbol *s, t_object *v, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that passes a single `t_object*` as an argument.*
  - `t_max_err object_method_char_array (t_object *x, t_symbol *s, long ac, unsigned char *av, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that passes an array of char values as an argument.*
  - `t_max_err object_method_long_array (t_object *x, t_symbol *s, long ac, t_atom_long *av, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that passes an array of long integers values as an argument.*
  - `t_max_err object_method_float_array (t_object *x, t_symbol *s, long ac, float *av, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that passes an array of 32bit floats values as an argument.*
  - `t_max_err object_method_double_array (t_object *x, t_symbol *s, long ac, double *av, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that passes an array of 64bit float values as an argument.*
  - `t_max_err object_method_sym_array (t_object *x, t_symbol *s, long ac, t_symbol **av, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that passes an array of `t_symbol*` values as an argument.*
  - `t_max_err object_method_obj_array (t_object *x, t_symbol *s, long ac, t_object **av, t_atom *rv)`

*Convenience wrapper for `object_method_typed()` that passes an array of `t_object*` values as an argument.*
  - `void object_openhelp (t_object *x)`

*Open the help patcher for a given instance of an object.*
  - `void object_openrefpage (t_object *x)`

*Open the reference page for a given instance of an object.*
  - `void object_openquery (t_object *x)`

*Open a search in the file browser for files with the name of the given object.*
  - `void classname_openhelp (char *classname)`

*Open the help patcher for a given object class name.*
  - `void classname_openrefpage (char *classname)`

*Open the reference page for a given object class name.*
  - `void classname_openquery (char *classname)`

*Open a search in the file browser for files with the name of the given class.*
  - `t_object * newobject_sprintf (t_object *patcher, C74_CONST char *fmt,...)`

*Create a new object in a specified patcher with values using a combination of attribute and sprintf syntax.*

- `t_object * newobject_fromboxtext (t_object *patcher, const char *text)`  
*Create an object from the passed in text.*
- `t_object * newobject_fromdictionary (t_object *patcher, t_dictionary *d)`  
*Place a new object into a patcher.*

### 38.39.1 Detailed Description

#### See also

<http://www.cycling74.com/twiki/bin/view/ProductDocumentation/JitterSdk/ObjectModel>

<http://www.cycling74.com/twiki/bin/view/ProductDocumentation/JitterSdk/RegNotify>

### 38.39.2 Macro Definition Documentation

#### 38.39.2.1 `object_method_direct`

```
#define object_method_direct (
    rt,
    sig,
    x,
    s,
    ... )
```

do a strongly typed direct call to a method of an object

#### Parameters

<code>rt</code>	The type of the return value (double, void*, void...)
<code>sig</code>	the actual signature of the function in brackets ! something like ( <code>t_object *</code> , double, long)
<code>x</code>	The object where the method we want to call will be looked for, it will also always be the first argument to the function call
<code>s</code>	The message selector
<code>...</code>	Any arguments to the call, the first one will always be the object ( <code>x</code> )

#### Returns

will return anything that the called function returns, typed by (`rt`)

### Remarks

Example: To call the function identified by `getcolorat` on the object `pwindow` which is declared like: `t_jrgba`

```
pwwindow_getcolorat(t_object *window, double x, double y)
double x = 44.73;
double y = 79.21;
t_object *pwindow;
t_jrgba result = object_method_direct(t_jrgba, (t_object *, double, double), pwindow, gensym("getcolorat"),
x, y);
```

## 38.39.3 Enumeration Type Documentation

### 38.39.3.1 anonymous enum

anonymous enum

Maximum number of arguments that can be passed as a typed-list rather than using [A\\_GIMME](#).

It is generally recommended to use [A\\_GIMME](#).

## 38.39.4 Function Documentation

### 38.39.4.1 classname\_openhelp()

```
void classname_openhelp (
    char * classname )
```

Open the help patcher for a given object class name.

#### Parameters

<code>classname</code>	The class name for which to open the help patcher.
------------------------	--

### 38.39.4.2 classname\_openquery()

```
void classname_openquery (
    char * classname )
```

Open a search in the file browser for files with the name of the given class.

**Parameters**

<i>classname</i>	The class name for which to query.
------------------	------------------------------------

**38.39.4.3 classname\_openrefpage()**

```
void classname_openrefpage (
    char * classname )
```

Open the reference page for a given object class name.

**Parameters**

<i>classname</i>	The class name for which to open the reference page.
------------------	--

**38.39.4.4 newobject\_fromboxtext()**

```
t_object* newobject_fromboxtext (
    t_object * patcher,
    const char * text )
```

Create an object from the passed in text.

The passed in text is in the same format as would be typed into an object box. It can be used for UI objects or text objects so this is the simplest way to create objects from C.

**Parameters**

<i>patcher</i>	An instance of a patcher object.
<i>text</i>	The text as if typed into an object box.

**Returns**

A pointer to the newly created object instance, or NULL if creation of the object fails.

**See also**

[newobject\\_sprintf\(\)](#)

### 38.39.4.5 newobject\_fromdictionary()

```
t_object* newobject_fromdictionary (
    t_object * patcher,
    t_dictionary * d )
```

Place a new object into a patcher.

The new object will be created based on a specification contained in a [Dictionary](#).

Create a new dictionary populated with values using a combination of attribute and sprintf syntax.

#### Parameters

<i>patcher</i>	An instance of a patcher object.
<i>d</i>	A dictionary containing an object specification.

#### Returns

A pointer to the newly created object instance, or NULL if creation of the object fails.

#### Remarks

Max attribute syntax is used to define key-value pairs. For example,  
`"@key1 value @key2 another_value"`

The example below creates a new object that in a patcher whose object pointer is stored in a variable called "aPatcher".

```
t_dictionary *d;
t_object *o;
char text[4];
strncpy_zero(text, "foo", 4);
d = dictionary_sprintf("@maxclass comment @varname _name \
    @text \"%s\" @patching_rect %.2f %.2f %.2f %.2f \
    @fontsize %f @textcolor %f %f %f 1.0 \
    @fontname %s @bgcolor 0.001 0.001 0.001 0.",
    text, 20.0, 20.0, 200.0, 24.0,
    18, 0.9, 0.9, 0.9, "Arial");
o = newobject_fromdictionary(aPatcher, d);
```

#### See also

- [newobject\\_sprintf\(\)](#)
- [newobject\\_fromdictionary\(\)](#)
- [atom\\_setparse\(\)](#)

### 38.39.4.6 newobject\_sprintf()

```
t_object* newobject_sprintf (
    t_object * patcher,
    C74_CONST char * fmt,
    ... )
```

Create a new object in a specified patcher with values using a combination of attribute and sprintf syntax.

**Parameters**

<i>patcher</i>	An instance of a patcher object.
<i>fmt</i>	An sprintf-style format string specifying key-value pairs with attribute nomenclature.
...	One or more arguments which are to be substituted into the format string.

**Returns**

A pointer to the newly created object instance, or NULL if creation of the object fails.

**Remarks**

Max attribute syntax is used to define key-value pairs. For example,  
 "@key1 value @key2 another\_value"

The example below creates a new object that in a patcher whose object pointer is stored in a variable called "aPatcher".

```
t_object *my_comment;
char text[4];
strncpy_zero(text, "foo", 4);
my_comment = newobject_sprintf(aPatcher, "@maxclass comment @varname _name \
@text \"%s\" @patching_rect %.2f %.2f %.2f %.2f \
@fontsize %f @textcolor %f %f %f 1.0 \
@fontname %s @bgcolor 0.001 0.001 0.001 0.",
text, 20.0, 20.0, 200.0, 24.0,
18, 0.9, 0.9, 0.9, "Arial");
```

**See also**

[dictionary\\_sprintf\(\)](#)  
[newobject\\_fromdictionary\(\)](#)  
[atom\\_setparse\(\)](#)

**38.39.4.7 object\_alloc()**

```
void* object_alloc (
    t_class * c )
```

Allocates the memory for an instance of an object class and initialize its object header.

It is used like the traditional function newobject, inside of an object's new method, but its use is required with obex-class objects.

**Parameters**

<i>c</i>	The class pointer, returned by <a href="#">class_new()</a>
----------	--

**Returns**

This function returns a new instance of an object class if successful, or NULL if unsuccessful.

**38.39.4.8 object\_attach()**

```
void* object_attach (
    t_symbol * name_space,
    t_symbol * s,
    void * x )
```

Attaches a client to a registered object.

Once attached, the object will receive notifications sent from the registered object (via the [object\\_notify\(\)](#) function), if it has a `notify` method defined and implemented.

**Parameters**

<i>name_space</i>	The namespace of the registered object. This should be the same value used in <a href="#">object_register()</a> to register the object. If you don't know the registered object's namespace, the <a href="#">object_findregisteredbyptr()</a> function can be used to determine it.
<i>s</i>	The name of the registered object in the namespace. If you don't know the name of the registered object, the <a href="#">object_findregisteredbyptr()</a> function can be used to determine it.
<i>x</i>	The client object to attach. Generally, this is the pointer to your Max object.

**Returns**

This function returns a pointer to the registered object (to the object referred to by the combination of `name_space` and `s` arguments) if successful, or NULL if unsuccessful.

**Remarks**

You should not attach an object to itself if the object is a UI object. UI objects automatically register and attach to themselves in [jbox\\_new\(\)](#).

**See also**

- [object\\_notify\(\)](#)
- [object\\_detach\(\)](#)
- [object\\_attach\\_byptr\(\)](#)
- [object\\_register\(\)](#)

Referenced by `jit_object_attach()`.

### 38.39.4.9 `object_attach_byptr()`

```
t_max_err object_attach_byptr (
    void * x,
    void * registeredobject )
```

Attaches a client to a registered object.

Unlike [object\\_attach\(\)](#), the client is specified by providing a pointer to that object rather than the registered name of that object.

Once attached, the object will receive notifications sent from the registered object (via the [object\\_notify\(\)](#) function), if it has a `notify` method defined and implemented.

#### Parameters

<code>x</code>	The attaching client object. Generally, this is the pointer to your Max object.
<code>registeredobject</code>	A pointer to the registered object to which you wish to attach.

#### Returns

A Max error code.

#### Remarks

You should not attach an object to itself if the object is a UI object. UI objects automatically register and attach to themselves in [jbox\\_new\(\)](#).

#### See also

- [object\\_notify\(\)](#)
- [object\\_detach\(\)](#)
- [object\\_attach\(\)](#)
- [object\\_register\(\)](#)
- [object\\_attach\\_byptr\\_register\(\)](#)

### 38.39.4.10 `object_attach_byptr_register()`

```
t_max_err object_attach_byptr_register (
    void * x,
    void * object_to_attach,
    t_symbol * reg_name_space )
```

A convenience function wrapping [object\\_register\(\)](#) and [object\\_attach\\_byptr\(\)](#).

**Parameters**

<i>x</i>	The attaching client object. Generally, this is the pointer to your Max object.
<i>object_to_attach</i>	A pointer to the object to which you wish to register and then to which to attach.
<i>reg_name_space</i>	The namespace in which to register the object_to_attach.

**Returns**

A Max error code.

**See also**

[object\\_register\(\)](#)  
[object\\_attach\\_byptr\(\)](#)

**38.39.4.11 object\_attr\_touch()**

```
t_max_err object_attr_touch (
    t_object * x,
    t_symbol * attrname )
```

Mark an attribute as being touched by some code not from the attribute setter.

This will notify clients that the attribute has changed.

**Parameters**

<i>x</i>	The object whose attribute has been changed
<i>attrname</i>	The attribute name

**Returns**

A Max error code

**38.39.4.12 object\_attr\_touch\_parse()**

```
t_max_err object_attr_touch_parse (
    t_object * x,
    char * attrnames )
```

Mark one or more attributes as being touched by some code not from the attribute setter.

This will notify clients that the attributes have changed. Utility to call [object\\_attr\\_touch\(\)](#) for several attributes

**Parameters**

<i>x</i>	The object whose attribute has been changed
<i>attrnames</i>	The attribute names as a space separated string

**Returns**

A Max error code

**38.39.4.13 object\_class()**

```
t_class* object_class (
    void * x )
```

Determines the class of a given object.

**Parameters**

<i>x</i>	The object to test
----------	--------------------

**Returns**

This function returns the `t_class` \* of the object's class, if successful, or NULL, if unsuccessful.

Referenced by `jit_object_class()`, and `max_jit_obex_new()`.

**38.39.4.14 object\_classname()**

```
t_symbol* object_classname (
    void * x )
```

Retrieves an object instance's class name.

**Parameters**

<i>x</i>	The object instance whose class name is being queried
----------	---

**Returns**

The classname, or NULL if unsuccessful.

Referenced by `jit_object_classname()`.

#### 38.39.4.15 `object_classname_compare()`

```
long object_classname_compare (
    void * x,
    t_symbol * name )
```

Determines if a particular object is an instance of a given class.

##### Parameters

<code>x</code>	The object to test
<code>name</code>	The name of the class to test this object against

##### Returns

This function returns 1 if the object is an instance of the named class. Otherwise, 0 is returned.

##### Remarks

For instance, to determine whether an unknown object pointer is a pointer to a print object, one would call:

```
long isprint = object_classname_compare(x, gensym("print"));
```

#### 38.39.4.16 `object_detach()`

```
t_max_err object_detach (
    t_symbol * name_space,
    t_symbol * s,
    void * x )
```

Detach a client from a registered object.

##### Parameters

<code>name_space</code>	The namespace of the registered object. This should be the same value used in <code>object_register()</code> to register the object. If you don't know the registered object's namespace, the <code>object_findregisteredbyptr()</code> function can be used to determine it.
<code>s</code>	The name of the registered object in the namespace. If you don't know the name of the registered object, the <code>object_findregisteredbyptr()</code> function can be used to determine it.
<code>x</code>	The client object to attach. Generally, this is the pointer to your Max object.

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `jit_object_detach()`.

**38.39.4.17 object\_detach\_byptr()**

```
t_max_err object_detach_byptr (
    void * x,
    void * registeredobject )
```

Detach a client from a registered object.

**Parameters**

<code>x</code>	The attaching client object. Generally, this is the pointer to your Max object.
<code>registeredobject</code>	The object from which to detach.

**Returns**

A Max error code.

**See also**

[object\\_detach\(\)](#)  
[object\\_attach\\_byptr\(\)](#)

**38.39.4.18 object\_dictionaryarg()**

```
t_dictionary* object_dictionaryarg (
    long ac,
    t_atom * av )
```

Retrieve a pointer to a dictionary passed in as an atom argument.

Use this function when working with classes that have dictionary constructors to fetch the dictionary.

**Parameters**

<code>ac</code>	The number of atoms.
<code>av</code>	A pointer to the first atom in the array.

**Returns**

The dictionary retrieved from the atoms.

**See also**

[attr\\_dictionary\\_process\(\)](#)

**38.39.4.19 object\_findregistered()**

```
void* object_findregistered (
    t_symbol * name_space,
    t_symbol * s )
```

Determines a registered object's pointer, given its namespace and name.

**Parameters**

<i>name_space</i>	The namespace of the registered object
<i>s</i>	The name of the registered object in the namespace

**Returns**

This function returns the pointer of the registered object, if successful, or NULL, if unsuccessful.

Referenced by [jit\\_object\\_findregistered\(\)](#).

**38.39.4.20 object\_findregisteredbyptr()**

```
t_max_err object_findregisteredbyptr (
    t_symbol ** name_space,
    t_symbol ** s,
    void * x )
```

Determines the namespace and/or name of a registered object, given the object's pointer.

**Parameters**

<i>name_space</i>	Pointer to a <a href="#">t_symbol</a> *, to receive the namespace of the registered object
<i>s</i>	Pointer to a <a href="#">t_symbol</a> *, to receive the name of the registered object within the namespace
<i>x</i>	Pointer to the registered object

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `jit_object_findregisteredbyptr()`.

**38.39.4.21 object\_free()**

```
t_max_err object_free (
    void * x )
```

Call the free function and release the memory for an instance of an internal object class previously instantiated using `object_new()`, `object_new_typed()` or other new-style object constructor functions (e.g.

`hashtab_new()`). It is, at the time of this writing, a wrapper for the traditional function `freeobject()`, but its use is suggested with obex-class objects.

**Parameters**

<code>x</code>	The pointer to the object to be freed.
----------------	--

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `jit_object_free()`.

**38.39.4.22 object\_getmethod()**

```
method object_getmethod (
    void * x,
    t_symbol * s )
```

Retrieves an object's #method for a particular message selector.

**Parameters**

<code>x</code>	The object whose method is being queried
<code>s</code>	The message selector

**Returns**

This function returns the #method if successful, or [method\\_false\(\)](#) if unsuccessful.

Referenced by [jit\\_object\\_getmethod\(\)](#).

**38.39.4.23 object\_getvalueof()**

```
t_max_err object_getvalueof (
    void * x,
    long * ac,
    t_atom ** av )
```

Retrieves the value of an object which supports the `getvalueof/setvalueof` interface.

See part 2 of the pattr SDK for more information on this interface.

**Parameters**

<code>x</code>	The object whose value is of interest
<code>ac</code>	Pointer to a long variable to receive the count of arguments in <code>av</code> . The long variable itself should be set to 0 previous to calling this function.
<code>av</code>	Pointer to a <code>t_atom *</code> , to receive object data. The <code>t_atom *</code> itself should be set to NULL previous to calling this function.

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

**Remarks**

Calling the [object\\_getvalueof\(\)](#) function allocates memory for any data it returns. It is the developer's responsibility to free it, using the [freebytes\(\)](#) function.

Developers wishing to design objects which will support this function being called on them must define and implement a special method, `getvalueof`, like so:

```
class_admmethod(c, (method)myobject_getvalueof, "getvalueof", A_CANT, 0);
```

The `getvalueof` method should be prototyped as:

```
t_max_err myobject_getvalueof(t_myobj *x, long *ac, t_atom **av);
```

And implemented, generally, as:

```
t_max_err myobj_getvalueof(t_myobj *x, long *ac, t_atom **av)
{
    if (ac && av) {
        if (*ac && *av) {
            // memory has been passed in; use it.
        } else {
            // allocate enough memory for your data
            *av = (t_atom *)getbytes(sizeof(t_atom));
        }
        *ac = 1; // our data is a single floating point value
        atom_setfloat(*av, x->objvalue);
    }
}
```

```

    }
    return MAX_ERR_NONE;
}
@remark      By convention, and to permit the interoperability of objects using the obex API,
             developers should allocate memory in their <tt>getvalueof</tt> methods using the getbytes()
             function.

```

Referenced by max\_jit\_obex\_gimmeback\_dumpout().

### 38.39.4.24 object\_method()

```
void* object_method (
    void * x,
    t_symbol * s,
    ... )
```

Sends an untyped message to an object.

There are some caveats to its use, however, particularly for 64-bit architectures. [object\\_method\\_direct\(\)](#) should be used in cases where floating-point or other non-integer types are being passed on the stack or in return values.

#### Parameters

<i>x</i>	The object that will receive the message
<i>s</i>	The message selector
...	Any arguments to the message

#### Returns

If the receiver object can respond to the message, [object\\_method\(\)](#) returns the result. Otherwise, the function will return 0.

#### Remarks

**Example:** To send the message bang to the object bang\_me:

```
void *bang_result;
bang_result = object_method(bang_me, gensym("bang"));
```

Referenced by jit\_attr\_symcompare().

### 38.39.4.25 object\_method\_char()

```
t_max_err object_method_char (
    t_object * x,
    t_symbol * s,
    unsigned char v,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes a single char as an argument.

**Parameters**

<i>x</i>	The object to which the message will be sent.
<i>s</i>	The name of the method to call on the object.
<i>v</i>	An argument to pass to the method.
<i>rv</i>	The address of an atom to hold a return value.

**Returns**

A Max error code.

**See also**

[object\\_method\\_typed\(\)](#)

**38.39.4.26 object\_method\_char\_array()**

```
t_max_err object_method_char_array (
    t_object * x,
    t_symbol * s,
    long ac,
    unsigned char * av,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes an array of char values as an argument.

**Parameters**

<i>x</i>	The object to which the message will be sent.
<i>s</i>	The name of the method to call on the object.
<i>ac</i>	The number of arguments to pass to the method.
<i>av</i>	The address of the first of the array of arguments to pass to the method.
<i>rv</i>	The address of an atom to hold a return value.

**Returns**

A Max error code.

**See also**

[object\\_method\\_typed\(\)](#)

### 38.39.4.27 object\_method\_double()

```
t_max_err object_method_double (
    t_object * x,
    t_symbol * s,
    double v,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes a single 64bit float as an argument.

#### Parameters

<i>x</i>	The object to which the message will be sent.
<i>s</i>	The name of the method to call on the object.
<i>v</i>	An argument to pass to the method.
<i>rv</i>	The address of an atom to hold a return value.

#### Returns

A Max error code.

#### See also

[object\\_method\\_typed\(\)](#)

### 38.39.4.28 object\_method\_double\_array()

```
t_max_err object_method_double_array (
    t_object * x,
    t_symbol * s,
    long ac,
    double * av,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes an array of 64bit float values as an argument.

#### Parameters

<i>x</i>	The object to which the message will be sent.
<i>s</i>	The name of the method to call on the object.
<i>ac</i>	The number of arguments to pass to the method.
<i>av</i>	The address of the first of the array of arguments to pass to the method.
<i>rv</i>	The address of an atom to hold a return value.

**Returns**

A Max error code.

**See also**

[object\\_method\\_typed\(\)](#)

**38.39.4.29 object\_method\_float()**

```
t_max_err object_method_float (
    t_object * x,
    t_symbol * s,
    float v,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes a single 32bit float as an argument.

**Parameters**

<i>x</i>	The object to which the message will be sent.
<i>s</i>	The name of the method to call on the object.
<i>v</i>	An argument to pass to the method.
<i>rv</i>	The address of an atom to hold a return value.

**Returns**

A Max error code.

**See also**

[object\\_method\\_typed\(\)](#)

**38.39.4.30 object\_method\_float\_array()**

```
t_max_err object_method_float_array (
    t_object * x,
    t_symbol * s,
    long ac,
    float * av,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes an array of 32bit floats values as an argument.

**Parameters**

<i>x</i>	The object to which the message will be sent.
<i>s</i>	The name of the method to call on the object.
<i>ac</i>	The number of arguments to pass to the method.
<i>av</i>	The address of the first of the array of arguments to pass to the method.
<i>rv</i>	The address of an atom to hold a return value.

**Returns**

A Max error code.

**See also**

[object\\_method\\_typed\(\)](#)

**38.39.4.31 object\_method\_format()**

```
t_max_err object_method_format (
    t_object * x,
    t_symbol * s,
    t_atom * rv,
    C74_CONST char * fmt,
    ...
)
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that uses [atom\\_setformat\(\)](#) to define the arguments.

**Parameters**

<i>x</i>	The object to which the message will be sent.
<i>s</i>	The name of the method to call on the object.
<i>rv</i>	The address of an atom to hold a return value.
<i>fmt</i>	An sprintf-style format string specifying values for the atoms.
...	One or more arguments which are to be substituted into the format string.

**Returns**

A Max error code.

**See also**

[object\\_method\\_typed\(\)](#)  
[atom\\_setformat\(\)](#)

### 38.39.4.32 object\_method\_long()

```
t_max_err object_method_long (
    t_object * x,
    t_symbol * s,
    long v,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes a single long integer as an argument.

#### Parameters

<i>x</i>	The object to which the message will be sent.
<i>s</i>	The name of the method to call on the object.
<i>v</i>	An argument to pass to the method.
<i>rv</i>	The address of an atom to hold a return value.

#### Returns

A Max error code.

#### See also

[object\\_method\\_typed\(\)](#)

### 38.39.4.33 object\_method\_long\_array()

```
t_max_err object_method_long_array (
    t_object * x,
    t_symbol * s,
    long ac,
    t_atom_long * av,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes an array of long integers values as an argument.

#### Parameters

<i>x</i>	The object to which the message will be sent.
<i>s</i>	The name of the method to call on the object.
<i>ac</i>	The number of arguments to pass to the method.
<i>av</i>	The address of the first of the array of arguments to pass to the method.
<i>rv</i>	The address of an atom to hold a return value.

**Returns**

A Max error code.

**See also**

[object\\_method\\_typed\(\)](#)

**38.39.4.34 object\_method\_obj()**

```
t_max_err object_method_obj (
    t_object * x,
    t_symbol * s,
    t_object * v,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes a single `t_object*` as an argument.

**Parameters**

<code>x</code>	The object to which the message will be sent.
<code>s</code>	The name of the method to call on the object.
<code>v</code>	An argument to pass to the method.
<code>rv</code>	The address of an atom to hold a return value.

**Returns**

A Max error code.

**See also**

[object\\_method\\_typed\(\)](#)

**38.39.4.35 object\_method\_obj\_array()**

```
t_max_err object_method_obj_array (
    t_object * x,
    t_symbol * s,
    long ac,
    t_object ** av,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes an array of `t_object*` values as an argument.

**Parameters**

<i>x</i>	The object to which the message will be sent.
<i>s</i>	The name of the method to call on the object.
<i>ac</i>	The number of arguments to pass to the method.
<i>av</i>	The address of the first of the array of arguments to pass to the method.
<i>rv</i>	The address of an atom to hold a return value.

**Returns**

A Max error code.

**See also**

[object\\_method\\_typed\(\)](#)

**38.39.4.36 object\_method\_parse()**

```
t_max_err object_method_parse (
    t_object * x,
    t_symbol * s,
    C74_CONST char * parsestr,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that uses [atom\\_setparse\(\)](#) to define the arguments.

**Parameters**

<i>x</i>	The object to which the message will be sent.
<i>s</i>	The name of the method to call on the object.
<i>parsestr</i>	A C-string to parse into an array of atoms to pass to the method.
<i>rv</i>	The address of an atom to hold a return value.

**Returns**

A Max error code.

**See also**

[object\\_method\\_typed\(\)](#)  
[atom\\_setparse\(\)](#)

### 38.39.4.37 object\_method\_sym()

```
t_max_err object_method_sym (
    t_object * x,
    t_symbol * s,
    t_symbol * v,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes a single `t_symbol*` as an argument.

#### Parameters

<code>x</code>	The object to which the message will be sent.
<code>s</code>	The name of the method to call on the object.
<code>v</code>	An argument to pass to the method.
<code>rv</code>	The address of an atom to hold a return value.

#### Returns

A Max error code.

#### See also

[object\\_method\\_typed\(\)](#)

### 38.39.4.38 object\_method\_sym\_array()

```
t_max_err object_method_sym_array (
    t_object * x,
    t_symbol * s,
    long ac,
    t_symbol ** av,
    t_atom * rv )
```

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes an array of `t_symbol*` values as an argument.

#### Parameters

<code>x</code>	The object to which the message will be sent.
<code>s</code>	The name of the method to call on the object.
<code>ac</code>	The number of arguments to pass to the method.
<code>av</code>	The address of the first of the array of arguments to pass to the method.
<code>rv</code>	The address of an atom to hold a return value.

**Returns**

A Max error code.

**See also**

[object\\_method\\_typed\(\)](#)

**38.39.4.39 object\_method\_typed()**

```
t_max_err object_method_typed (
    void * x,
    t_symbol * s,
    long ac,
    t_atom * av,
    t_atom * rv )
```

Sends a type-checked message to an object.

**Parameters**

<i>x</i>	The object that will receive the message
<i>s</i>	The message selector
<i>ac</i>	Count of message arguments in <i>av</i>
<i>av</i>	Array of <i>t_atoms</i> ; the message arguments
<i>rv</i>	Return value of function, if available

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

**Remarks**

If the receiver object can respond to the message, [object\\_method\\_typed\(\)](#) returns the result in *rv*. Otherwise, *rv* will contain an [A\\_NOTHING](#) atom.

Referenced by [jit\\_object\\_method\\_typed\(\)](#), [max\\_jit\\_attr\\_args\(\)](#), [max\\_jit\\_obex\\_gimmeback\(\)](#), and [max\\_jit\\_obex\\_gimmeback\\_dumpout\(\)](#).

### 38.39.4.40 object\_method\_typedfun()

```
t_max_err object_method_typedfun (
    void * x,
    t_messlist * mp,
    t_symbol * s,
    long ac,
    t_atom * av,
    t_atom * rv )
```

Currently undocumented.

#### Parameters

<i>x</i>	The object that will receive the message
<i>mp</i>	Undocumented
<i>s</i>	The message selector
<i>ac</i>	Count of message arguments in <i>av</i>
<i>av</i>	Array of <i>t_atoms</i> ; the message arguments
<i>rv</i>	Return value of function, if available

#### Returns

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

#### Remarks

If the receiver object can respond to the message, [object\\_method\\_typedfun\(\)](#) returns the result in *rv*. Otherwise, *rv* will contain an [A NOTHING](#) atom.

### 38.39.4.41 object\_new()

```
void* object_new (
    t_symbol * name_space,
    t_symbol * classname,
    ... )
```

Allocates the memory for an instance of an object class and initialize its object header *internal to Max*.

It is used similarly to the traditional function [newinstance\(\)](#), but its use is required with obex-class objects.

**Parameters**

<i>name_space</i>	The desired object's name space. Typically, either the constant <a href="#">CLASS_BOX</a> , for obex classes which can instantiate inside of a Max patcher (e.g. boxes, UI objects, etc.), or the constant <a href="#">CLASS_NOBOX</a> , for classes which will only be used internally. Developers can define their own name spaces as well, but this functionality is currently undocumented.
<i>classname</i>	The name of the class of the object to be created
...	Any arguments expected by the object class being instantiated

**Returns**

This function returns a new instance of the object class if successful, or NULL if unsuccessful.

**38.39.4.42 [object\\_new\\_typed\(\)](#)**

```
void* object_new_typed (
    t_symbol * name_space,
    t_symbol * classname,
    long ac,
    t_atom * av )
```

Allocates the memory for an instance of an object class and initialize its object header *internal to Max*.

It is used similarly to the traditional function [newinstance\(\)](#), but its use is required with obex-class objects. The [object\\_new\\_typed\(\)](#) function differs from [object\\_new\(\)](#) by its use of an atom list for object arguments—in this way, it more resembles the effect of typing something into an object box from the Max interface.

**Parameters**

<i>name_space</i>	The desired object's name space. Typically, either the constant <a href="#">CLASS_BOX</a> , for obex classes which can instantiate inside of a Max patcher (e.g. boxes, UI objects, etc.), or the constant <a href="#">CLASS_NOBOX</a> , for classes which will only be used internally. Developers can define their own name spaces as well, but this functionality is currently undocumented.
<i>classname</i>	The name of the class of the object to be created
<i>ac</i>	Count of arguments in <i>av</i>
<i>av</i>	Array of <i>t_atoms</i> ; arguments to the class's instance creation function.

**Returns**

This function returns a new instance of the object class if successful, or NULL if unsuccessful.

### 38.39.4.43 object\_notify()

```
t_max_err object_notify (
    void * x,
    t_symbol * s,
    void * data )
```

Broadcast a message (with an optional argument) from a registered object to any attached client objects.

#### Parameters

<i>x</i>	Pointer to the registered object
<i>s</i>	The message to send
<i>data</i>	An optional argument which will be passed with the message. Sets this argument to NULL if it will be unused.

#### Returns

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

#### Remarks

In order for client objects to receive notifications, they must define and implement a special method, `notify`, like so:

```
class_addmethod(c, (method)myobject_notify, "notify", A_CANT, 0);
```

The `notify` method should be prototyped as:

```
void myobject_notify(t_myobject *x, t_symbol *s, t_symbol *msg, void *sender, void *data);
```

where *x* is the pointer to the receiving object, *s* is the name of the sending (registered) object in its namespace, *msg* is the sent message, *sender* is the pointer to the sending object, and *data* is an optional argument sent with the message. This value corresponds to the *data* argument in the `object_notify()` method.

Referenced by `jit_object_notify()`, `max_jit_attr_set()`, and `max_jit_obex_attr_set()`.

### 38.39.4.44 object\_obex\_dumpout()

```
void object_obex_dumpout (
    void * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Sends data from the object's dumpout outlet.

The dumpout outlet is stored in the obex using the `object_obex_store()` function (see above). It is used approximately like `outlet_anything()`.

**Parameters**

<i>x</i>	The object pointer. This function should only be called on instantiated objects (i.e. in the <code>new</code> method or later), not directly on classes (i.e. in <code>main()</code> ).
<i>s</i>	The message selector <code>t_symbol *</code>
<i>argc</i>	Number of elements in the argument list in <code>argv</code>
<i>argv</i>	<code>t_atoms</code> constituting the message arguments

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

Referenced by `ext_main()`.

**38.39.4.45 `object_obex_lookup()`**

```
t_max_err object_obex_lookup (
    void * x,
    t_symbol * key,
    t_object ** val )
```

Retrieves the value of a data stored in the obex.

**Parameters**

<i>x</i>	The object pointer. This function should only be called on instantiated objects (i.e. in the <code>new</code> method or later), not directly on classes (i.e. in <code>main()</code> ).
<i>key</i>	The symbolic name for the data to be retrieved
<i>val</i>	A pointer to a <code>t_object *</code> , to be filled with the data retrieved from the obex.

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

**Remarks**

By default, pointers to the object's containing patcher and box objects are stored in the obex, under the keys '#P' and '#B', respectively. To retrieve them, the developer could do something like the following:

```
void post_containers(t_obexobj **x)
{
    t_patcher *p;
    t_box *b;
    t_max_err err;
    err = object_obex_lookup(x, gensym("#P"), (t_object **) &p);
    err = object_obex_lookup(x, gensym("#B"), (t_object **) &b);
    post("my patcher is located at 0x%X", p);
    post("my box is located at 0x%X", b);
}
```

### 38.39.4.46 object\_obex\_store()

```
t_max_err object_obex_store (
    void * x,
    t_symbol * key,
    t_object * val )
```

Stores data in the object's obex.

#### Parameters

<i>x</i>	The object pointer. This function should only be called on instantiated objects (i.e. in the <code>new</code> method or later), not directly on classes (i.e. in <code>main()</code> ).
<i>key</i>	A symbolic name for the data to be stored
<i>val</i>	A <code>t_object</code> *, to be stored in the obex, referenced under the <code>key</code> .

#### Returns

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

#### Remarks

Most developers will need to use this function for the specific purpose of storing the dumpout outlet in the obex (the dumpout outlet is used by attributes to report data in response to 'get' queries). For this, the developer should use something like the following in the object's `new` method:

```
object_obex_store(x, _sym_dumpout, outlet_new(x, NULL));
```

### 38.39.4.47 object\_openhelp()

```
void object_openhelp (
    t_object * x )
```

Open the help patcher for a given instance of an object.

#### Parameters

<i>x</i>	The object instance for which to open the help patcher.
----------	---

#### 38.39.4.48 object\_openquery()

```
void object_openquery (
    t_object * x )
```

Open a search in the file browser for files with the name of the given object.

##### Parameters

x	The object instance for which to query.
---	---

#### 38.39.4.49 object\_openrefpage()

```
void object_openrefpage (
    t_object * x )
```

Open the reference page for a given instance of an object.

##### Parameters

x	The object instance for which to open the reference page.
---	---

#### 38.39.4.50 object\_register()

```
void* object_register (
    t_symbol * name_space,
    t_symbol * s,
    void * x )
```

Registers an object in a namespace.

##### Parameters

<i>name_space</i>	The namespace in which to register the object. The namespace can be any symbol. If the namespace does not already exist, it is created automatically.
<i>s</i>	The name of the object in the namespace. This name will be used by other objects to attach and detach from the registered object.
<i>x</i>	The object to register

**Returns**

The function returns a pointer to the registered object. Under some circumstances, `object_register` will *duplicate* the object, and return a pointer to the duplicate—the developer should not assume that the pointer passed in is the same pointer that has been registered. To be safe, the returned pointer should be stored and used with the `object_unregister()` function.

**Remarks**

You should not register an object if the object is a UI object. UI objects automatically register and attach to themselves in `jbox_new()`.

Referenced by `jit_object_register()`.

**38.39.4.51 object\_register\_getnames()**

```
t_max_err object_register_getnames (
    t_symbol * name_space,
    long * namecount,
    t_symbol *** names )
```

Returns all registered names in a namespace.

**Parameters**

<code>name_space</code>	Pointer to a <code>t_symbol</code> , the namespace to lookup names in
<code>namecount</code>	Pointer to a long, to receive the count of the registered names within the namespace
<code>names</code>	Pointer to a <code>t_symbol</code> **, to receive the allocated names. This pointer should be freed after use

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in "ext\_obex.h" if unsuccessful.

**38.39.4.52 object\_setvalueof()**

```
t_max_err object_setvalueof (
    void * x,
    long ac,
    t_atom * av )
```

Sets the value of an object which supports the `getvalueof`/`setvalueof` interface.

**Parameters**

<i>x</i>	The object whose value is of interest
<i>ac</i>	The count of arguments in <i>av</i>
<i>av</i>	Array of <i>t_atoms</i> ; the new desired data for the object

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

**Remarks**

Developers wishing to design objects which will support this function being called on them must define and implement a special method, `setvalueof`, like so:

```
class_admmethod(c, (method)myobject_setvalueof, "setvalueof", A_CANT, 0);
```

The `setvalueof` method should be prototyped as:

```
t_max_err myobject_setvalueof(t_myobject *x, long *ac, t_atom **av);
```

And implemented, generally, as:

```
t_max_err myobject_setvalueof(t_myobject *x, long ac, t_atom *av)
{
    if (ac && av) {
        // simulate receipt of a float value
        myobject_float(x, atom_getfloat(av));
    }
    return MAX_ERR_NONE;
}
```

**38.39.4.53 object\_subscribe()**

```
void* object_subscribe (
    t_symbol * name_space,
    t_symbol * s,
    t_symbol * classname,
    void * x )
```

Subscribes a client to wait for an object to register.

Upon registration, the object will attach. Once attached, the object will receive notifications sent from the registered object (via the `object_notify` function), if it has a `notify` method defined and implemented. See below for more information, in the reference for `object_notify`.

**Parameters**

<i>name_space</i>	The namespace of the registered object. This should be the same value used in <code>object_register</code> to register the object. If you don't know the registered object's namespace, the <code>object_findregisteredbyptr</code> function can be used to determine it.
<i>s</i>	The name of the registered object in the namespace. If you don't know the name of the registered object, the <code>object_findregisteredbyptr</code> function can be used to determine it.
<i>classname</i>	The classname of the registered object in the namespace to use as a filter. If NULL, then it will attach to any class of object.
<i>Cycling '74 x</i>	The client object to attach. Generally, this is the pointer to your Max object.

**Returns**

This function returns a pointer to the object if registered (to the object referred to by the combination of `name_space` and `s` arguments) if successful, or NULL if the object is not yet registered.

**38.39.4.54 `object_super_method()`**

```
void* object_super_method (
    t_object * x,
    t_symbol * s,
    ... )
```

Sends an untyped message to an object using superclass methods.

Uses a thread specific stack to ensure traversal up the class hierarchy.

**Parameters**

<code>x</code>	The object that will receive the message
<code>s</code>	The message selector
...	Any arguments to the message

**Returns**

If the receiver object can respond to the message, `object_method()` returns the result. Otherwise, the function will return 0.

**38.39.4.55 `object_this_method()`**

```
void* object_this_method (
    t_object * x,
    t_symbol * s,
    ... )
```

Sends an untyped message to an object, respects a thread specific class stack from `object_super_method()` calls.

**Parameters**

<code>x</code>	The object that will receive the message
<code>s</code>	The message selector
...	Any arguments to the message

**Returns**

If the receiver object can respond to the message, [object\\_method\(\)](#) returns the result. Otherwise, the function will return 0.

**38.39.4.56 object\_unregister()**

```
t_max_err object_unregister (
    void * x )
```

Removes a registered object from a namespace.

**Parameters**

x	The object to unregister. This should be the pointer returned from the <a href="#">object_register()</a> function.
---	--

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

Referenced by [jit\\_object\\_unregister\(\)](#).

**38.39.4.57 object\_unsubscribe()**

```
t_max_err object_unsubscribe (
    t_symbol * name_space,
    t_symbol * s,
    t_symbol * classname,
    void * x )
```

Unsubscribe a client from a registered object, detaching if the object is registered.

**Parameters**

<i>name_space</i>	The namespace of the registered object. This should be the same value used in <a href="#">object_register</a> to register the object. If you don't know the registered object's namespace, the <a href="#">object_findregisteredbyptr</a> function can be used to determine it.
<i>s</i>	The name of the registered object in the namespace. If you don't know the name of the registered object, the <a href="#">object_findregisteredbyptr</a> function can be used to determine it.
<i>classname</i>	The classname of the registered object in the namespace to use as a filter. Currently unused for unsubscribe.
<i>x</i>	The client object to detach. Generally, this is the pointer to your Max object.

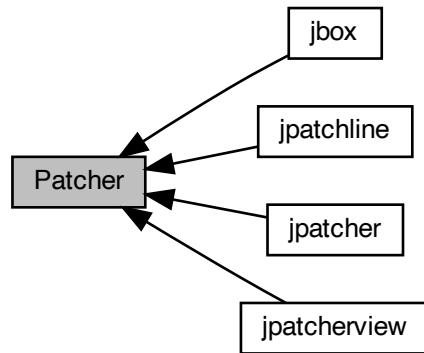
**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in "ext\_obex.h" if unsuccessful.

## 38.40 Patcher

Max's patcher represents a graph of objects that communicate with messages.

Collaboration diagram for Patcher:



## Modules

- [jpatcher](#)  
*The patcher.*
- [jbox](#)  
*A box in the patcher.*
- [jpatchline](#)  
*A patch cord.*
- [jpatcherview](#)  
*A view of a patcher.*

## Data Structures

- [struct t\\_jbox](#)  
*The `t_jbox` struct provides the header for a Max user-interface object.*

## Typedefs

- `typedef t_object t_box`  
*A box.*

## Enumerations

- `enum { PI_DEEP, PI_REQUIREFIRSTIN, PI_WANTBOX, PI_SKIPGEN, PI_WANTPATCHER }`  
*patcher iteration flags*

## Variables

- `BEGIN_USING_C_LINKAGE typedef t_object t_patcher`  
*A patcher.*

### 38.40.1 Detailed Description

Max's patcher represents a graph of objects that communicate with messages.

This is the public interface to the jpatcher – the new patcher object in Max 5. The jpatcher is fully controllable via obex attributes and methods.

The jpatcher\_api.h header defines constants, enumerations, symbols, structs, and functions for working with the jpatcher. It also includes utility functions for getting/setting attributes and for calling methods. These utilities are just wrapping the obex interface and thus loosely connect your code to the jpatcher implementation.

Finally methods are defined for implementing your own boxes.

### 38.40.2 Typedef Documentation

#### 38.40.2.1 t\_box

`typedef t_object t_box`

A box.

As of Max 5, the box struct is opaque. Messages can be sent to a box using `object_method()` or `object_method_typed()`, or by using `Attributes` accessors.

### 38.40.3 Enumeration Type Documentation

#### 38.40.3.1 anonymous enum

`anonymous enum`

patcher iteration flags

## Enumerator

PI_DEEP	descend into subpatchers (not used by audio library)
PI_REQUIREFIRSTIN	if b->b_firstin is NULL, do not call function
PI_WANTBOX	instead, of b->b_firstin, pass b to function, whether or not b->b_firstin is NULL

**38.40.4 Variable Documentation****38.40.4.1 t\_patcher**

```
BEGIN_USING_C_LINKAGE typedef t_object t_patcher
```

A patcher.

As of Max 5, the patcher struct is opaque. Messages can be sent to a patcher using [object\\_method\(\)](#) or [object\\_method\\_typed\(\)](#), or by using [Attributes](#) accessors.

**38.41 jpatcher**

The patcher.

Collaboration diagram for jpatcher:



## Functions

- int `jpatcher_is_patcher (t_object *p)`  
*Determine if a `t_object*` is a patcher object.*
- `t_object * jpatcher_get_box (t_object *p)`  
*If a patcher is inside a box, return its box.*
- long `jpatcher_get_count (t_object *p)`  
*Determine the number of boxes in a patcher.*
- `t_max_err jpatcher_set_locked (t_object *p, char c)`  
*Lock or unlock a patcher.*
- char `jpatcher_get_presentation (t_object *p)`  
*Determine whether a patcher is currently in presentation mode.*
- `t_max_err jpatcher_set_presentation (t_object *p, char c)`  
*Set a patcher to presentation mode.*
- `t_object * jpatcher_get_firstobject (t_object *p)`  
*Get the first box in a patcher.*
- `t_object * jpatcher_get_lastobject (t_object *p)`  
*Get the last box in a patcher.*
- `t_object * jpatcher_get_firstline (t_object *p)`  
*Get the first line (patch-cord) in a patcher.*
- `t_object * jpatcher_get_firstview (t_object *p)`  
*Get the first view (jpatcherview) for a given patcher.*
- `t_symbol * jpatcher_get_title (t_object *p)`  
*Retrieve a patcher's title.*
- `t_max_err jpatcher_set_title (t_object *p, t_symbol *ps)`  
*Set a patcher's title.*
- `t_symbol * jpatcher_get_name (t_object *p)`  
*Retrieve a patcher's name.*
- `t_symbol * jpatcher_get_filepath (t_object *p)`  
*Retrieve a patcher's file path.*
- `t_symbol * jpatcher_get_filename (t_object *p)`  
*Retrieve a patcher's file name.*
- char `jpatcher_get_dirty (t_object *p)`  
*Determine whether a patcher's dirty bit has been set.*
- `t_max_err jpatcher_set_dirty (t_object *p, char c)`  
*Set a patcher's dirty bit.*
- char `jpatcher_get_bglocked (t_object *p)`  
*Determine whether a patcher's background layer is locked.*
- `t_max_err jpatcher_set_bglocked (t_object *p, char c)`  
*Set whether a patcher's background layer is locked.*
- char `jpatcher_get_bghidden (t_object *p)`  
*Determine whether a patcher's background layer is hidden.*
- `t_max_err jpatcher_set_bghidden (t_object *p, char c)`  
*Set whether a patcher's background layer is hidden.*
- char `jpatcher_get_fghidden (t_object *p)`  
*Determine whether a patcher's foreground layer is hidden.*
- `t_max_err jpatcher_set_fghidden (t_object *p, char c)`

- `t_max_err jpatcher_get_editing_bgcolor (t_object *p, t_jrgba *prgba)`  
*Set whether a patcher's foreground layer is hidden.*
- `t_max_err jpatcher_set_editing_bgcolor (t_object *p, t_jrgba *prgba)`  
*Retrieve a patcher's editing background color.*
- `t_max_err jpatcher_get_bgcolor (t_object *p, t_jrgba *prgba)`  
*Set a patcher's editing background color.*
- `t_max_err jpatcher_get_locked_bgcolor (t_object *p, t_jrgba *prgba)`  
*Retrieve a patcher's unlocked background color.*
- `t_max_err jpatcher_set_bgcolor (t_object *p, t_jrgba *prgba)`  
*Set a patcher's unlocked background color.*
- `t_max_err jpatcher_set_locked_bgcolor (t_object *p, t_jrgba *prgba)`  
*Set a patcher's locked background color.*
- `t_max_err jpatcher_get_gridsize (t_object *p, double *gridsizeX, double *gridsizeY)`  
*Retrieve a patcher's grid size.*
- `t_max_err jpatcher_set_gridsize (t_object *p, double gridsizeX, double gridsizeY)`  
*Set a patcher's grid size.*
- `void jpatcher_deleteobj (t_object *p, t_jbox *b)`  
*Delete an object that is in a patcher.*
- `t_object * jpatcher_get_parentpatcher (t_object *p)`  
*Given a patcher, return its parent patcher.*
- `t_object * jpatcher_get_toppatcher (t_object *p)`  
*Given a patcher, return the top-level patcher for the tree in which it exists.*
- `t_object * jpatcher_get_hubholder (t_object *p)`  
*Given a patcher, return the patcher that will be responsible for holding the parameter hub.*
- `t_max_err jpatcher_get_rect (t_object *p, t_rect *pr)`  
*Query a patcher to determine its location and size.*
- `t_max_err jpatcher_set_rect (t_object *p, t_rect *pr)`  
*Set a patcher's location and size.*
- `t_max_err jpatcher_get_defrect (t_object *p, t_rect *pr)`  
*Query a patcher to determine the location and dimensions of its window when initially opened.*
- `t_max_err jpatcher_set_defrect (t_object *p, t_rect *pr)`  
*Set a patcher's default location and size.*
- `t_symbol * jpatcher_uniqueboxname (t_object *p, t_symbol *classname)`  
*Generate a unique name for a box in patcher.*
- `t_symbol * jpatcher_get_default_fontname (t_object *p)`  
*Return the name of the default font used for new objects in a patcher.*
- `float jpatcher_get_default_fontsize (t_object *p)`  
*Return the size of the default font used for new objects in a patcher.*
- `long jpatcher_get_default_fontface (t_object *p)`  
*Return the index of the default font face used for new objects in a patcher.*
- `long jpatcher_get_fileversion (t_object *p)`  
*Return the file version of the patcher.*
- `long jpatcher_get_currentfileversion (void)`  
*Return the file version for any new patchers, e.g.*

### 38.41.1 Detailed Description

The patcher.

### 38.41.2 Function Documentation

#### 38.41.2.1 jpatcher\_deleteobj()

```
void jpatcher_deleteobj (
    t_object * p,
    t_jbox * b )
```

Delete an object that is in a patcher.

##### Parameters

<i>p</i>	The patcher.
<i>b</i>	The object box to delete.

#### 38.41.2.2 jpatcher\_get\_bgcolor()

```
t_max_err jpatcher_get_bgcolor (
    t_object * p,
    t_jrgba * prgba )
```

Retrieve a patcher's unlocked background color.

##### Parameters

<i>p</i>	The patcher to be queried.
<i>prgba</i>	The address of a valid <a href="#">t_jrgba</a> struct that will be filled-in with the current patcher color values.

##### Returns

A Max error code.

### 38.41.2.3 jpatcher\_get\_bghidden()

```
char jpatcher_get_bghidden (
    t_object * p )
```

Determine whether a patcher's background layer is hidden.

#### Parameters

<i>p</i>	The patcher to be queried.
----------	----------------------------

#### Returns

True if the background layer is hidden, otherwise false.

### 38.41.2.4 jpatcher\_get\_bglocked()

```
char jpatcher_get_bglocked (
    t_object * p )
```

Determine whether a patcher's background layer is locked.

#### Parameters

<i>p</i>	The patcher to be queried.
----------	----------------------------

#### Returns

True if the background layer is locked, otherwise false.

### 38.41.2.5 jpatcher\_get\_box()

```
t_object* jpatcher_get_box (
    t_object * p )
```

If a patcher is inside a box, return its box.

#### Parameters

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

A pointer to the box containing the patcher, otherwise NULL.

**38.41.2.6 jpatcher\_get\_count()**

```
long jpatcher_get_count (
    t_object * p )
```

Determine the number of boxes in a patcher.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

The number of boxes in the patcher.

**38.41.2.7 jpatcher\_get\_currentfileversion()**

```
long jpatcher_get_currentfileversion (
    void )
```

Return the file version for any new patchers, e.g.

the current version created by Max.

**Returns**

The file version number.

**38.41.2.8 jpatcher\_get\_default\_fontface()**

```
long jpatcher_get_default_fontface (
    t_object * p )
```

Return the index of the default font face used for new objects in a patcher.

**Parameters**

<i>p</i>	A pointer to a patcher instance.
----------	----------------------------------

**Returns**

The index of the default font face used for new objects in a patcher.

**38.41.2.9 jpatcher\_get\_default\_fontname()**

```
t_symbol* jpatcher_get_default_fontname (
    t_object * p )
```

Return the name of the default font used for new objects in a patcher.

**Parameters**

<i>p</i>	A pointer to a patcher instance.
----------	----------------------------------

**Returns**

The name of the default font used for new objects in a patcher.

**38.41.2.10 jpatcher\_get\_default\_fontsize()**

```
float jpatcher_get_default_fontsize (
    t_object * p )
```

Return the size of the default font used for new objects in a patcher.

**Parameters**

<i>p</i>	A pointer to a patcher instance.
----------	----------------------------------

**Returns**

The size of the default font used for new objects in a patcher.

**38.41.2.11 jpatcher\_get\_defrect()**

```
t_max_err jpatcher_get_defrect (
    t_object * p,
    t_rect * pr )
```

Query a patcher to determine the location and dimensions of its window when initially opened.

**Parameters**

<i>p</i>	A pointer to a patcher instance.
<i>pr</i>	The address of valid <code>t_rect</code> whose values will be filled-in upon return.

**Returns**

A Max error code.

**38.41.2.12 jpatcher\_get\_dirty()**

```
char jpatcher_get_dirty (
    t_object * p )
```

Determine whether a patcher's dirty bit has been set.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

True if the patcher is dirty, otherwise false.

**38.41.2.13 jpatcher\_get\_editing\_bgcolor()**

```
t_max_err jpatcher_get_editing_bgcolor (
    t_object * p,
    t_jrgba * prgba )
```

Retrieve a patcher's editing background color.

**Parameters**

<i>p</i>	The patcher to be queried.
<i>prgba</i>	The address of a valid <a href="#">t_jrgba</a> struct that will be filled-in with the current patcher color values.

**Returns**

A Max error code.

**38.41.2.14 jpatcher\_get\_fghidden()**

```
char jpatcher_get_fghidden (
    t_object * p )
```

Determine whether a patcher's foreground layer is hidden.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

True if the foreground layer is hidden, otherwise false.

**38.41.2.15 jpatcher\_get\_filename()**

```
t_symbol* jpatcher_get_filename (
    t_object * p )
```

Retrieve a patcher's file name.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

The patcher's file name.

**38.41.2.16 jpatcher\_get\_filepath()**

```
t_symbol* jpatcher_get_filepath (
    t_object * p )
```

Retrieve a patcher's file path.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

The patcher's file path.

**38.41.2.17 jpatcher\_get\_fileversion()**

```
long jpatcher_get_fileversion (
    t_object * p )
```

Return the file version of the patcher.

**Parameters**

<i>p</i>	A pointer to the patcher whose version number is desired.
----------	---

**Returns**

The file version number.

**38.41.2.18 jpatcher\_get\_firstline()**

```
t_object* jpatcher_get_firstline (
    t_object * p )
```

Get the first line (patch-cord) in a patcher.

All lines in a patcher are maintained internally in a [t\\_linklist](#). Use this function to begin traversing a patcher's lines.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

The first jpatchline in a patcher.

**38.41.2.19 [jpatcher\\_get\\_firstobject\(\)](#)**

```
t_object* jpatcher_get_firstobject (
    t_object * p )
```

Get the first box in a patcher.

All boxes in a patcher are maintained internally in a [t\\_linklist](#). Use this function together with [jbox\\_get\\_nextobject\(\)](#) to traverse a patcher.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

The first box in a patcher.

**See also**

[jbox\\_get\\_prevobject\(\)](#) [jbox\\_get\\_nextobject\(\)](#) [jpatcher\\_get\\_lastobject\(\)](#)

**38.41.2.20 [jpatcher\\_get\\_firstview\(\)](#)**

```
t_object* jpatcher_get_firstview (
    t_object * p )
```

Get the first view (jpatcherview) for a given patcher.

All views of a patcher are maintained internally as a [t\\_linklist](#). Use this function to begin traversing a patcher's views.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

The first view of a patcher.

**38.41.2.21 jpatcher\_get\_gridsize()**

```
t_max_err jpatcher_get_gridsize (
    t_object * p,
    double * gridsizeX,
    double * gridsizeY )
```

Retrieve a patcher's grid size.

**Parameters**

<i>p</i>	The patcher to be queried.
<i>gridsizeX</i>	The address of a double that will be set to the current horizontal grid spacing for the patcher.
<i>gridsizeY</i>	The address of a double that will be set to the current vertical grid spacing for the patcher.

**Returns**

A Max error code.

**38.41.2.22 jpatcher\_get\_hubholder()**

```
t_object* jpatcher_get_hubholder (
    t_object * p )
```

Given a patcher, return the patcher that will be responsible for holding the parameter hub.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

The patcher's parameter hub holder.

**38.41.2.23 jpatcher\_get\_lastobject()**

```
t_object* jpatcher_get_lastobject (
    t_object * p )
```

Get the last box in a patcher.

All boxes in a patcher are maintained internally in a [t\\_linklist](#). Use this function together with [jbox\\_get\\_prevobject\(\)](#) to traverse a patcher.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

The last box in a patcher.

**See also**

[jbox\\_get\\_prevobject\(\)](#) [jbox\\_get\\_nextobject\(\)](#) [jpatcher\\_get\\_firstobject\(\)](#)

**38.41.2.24 jpatcher\_get\_locked\_bgcolor()**

```
t_max_err jpatcher_get_locked_bgcolor (
    t_object * p,
    t_jrgba * prgba )
```

Retrieve a patcher's locked background color.

**Parameters**

<i>p</i>	The patcher to be queried.
<i>prgba</i>	The address of a valid <a href="#">t_jrgba</a> struct that will be filled-in with the current patcher color values.

**Returns**

A Max error code.

**38.41.2.25 jpatcher\_get\_name()**

```
t_symbol* jpatcher_get_name (
    t_object * p )
```

Retrieve a patcher's name.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

The patcher's name.

**38.41.2.26 jpatcher\_get\_parentpatcher()**

```
t_object* jpatcher_get_parentpatcher (
    t_object * p )
```

Given a patcher, return its parent patcher.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

The patcher's parent patcher, if there is one. If there is no parent patcher (this is a top-level patcher) then NULL is returned.

**38.41.2.27 jpatcher\_get\_presentation()**

```
char jpatcher_get_presentation (
    t_object * p )
```

Determine whether a patcher is currently in presentation mode.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

True if the patcher is in presentation mode, otherwise false.

**38.41.2.28 jpatcher\_get\_rect()**

```
t_max_err jpatcher_get_rect (
    t_object * p,
    t_rect * pr )
```

Query a patcher to determine its location and size.

**Parameters**

<i>p</i>	A pointer to a patcher instance.
<i>pr</i>	The address of valid <a href="#">t_rect</a> whose values will be filled-in upon return.

**Returns**

A Max error code.

**38.41.2.29 jpatcher\_get\_title()**

```
t_symbol* jpatcher_get_title (
    t_object * p )
```

Retrieve a patcher's title.

**Parameters**

<i>p</i>	The patcher to be queried.
----------	----------------------------

**Returns**

The patcher's title.

### 38.41.2.30 jpatcher\_get\_toppatcher()

```
t_object* jpatcher_get_toppatcher (
    t_object * p )
```

Given a patcher, return the top-level patcher for the tree in which it exists.

#### Parameters

<i>p</i>	The patcher to be queried.
----------	----------------------------

#### Returns

The patcher's top-level parent patcher.

### 38.41.2.31 jpatcher\_is\_patch()

```
int jpatcher_is_patch (
    t_object * p )
```

Determine if a `t_object*` is a patcher object.

#### Parameters

<i>p</i>	The object pointer to test.
----------	-----------------------------

#### Returns

Returns true if the object is a patcher, otherwise returns non-zero.

### 38.41.2.32 jpatcher\_set\_bgcolor()

```
t_max_err jpatcher_set_bgcolor (
    t_object * p,
    t_jrgba * prgba )
```

Set a patcher's unlocked background color.

#### Parameters

<i>p</i>	The patcher to be queried.
<i>prgba</i>	The address of a <code>t_jrgba</code> struct containing the new color to use.

**Returns**

A Max error code.

**38.41.2.33 jpatcher\_set\_bghidden()**

```
t_max_err jpatcher_set_bghidden (
    t_object * p,
    char c )
```

Set whether a patcher's background layer is hidden.

**Parameters**

<i>p</i>	The patcher whose dirty bit will be set.
<i>c</i>	Pass true to hide the patcher's background layer, otherwise pass false.

**Returns**

A Max error code.

**38.41.2.34 jpatcher\_set\_bglocked()**

```
t_max_err jpatcher_set_bglocked (
    t_object * p,
    char c )
```

Set whether a patcher's background layer is locked.

**Parameters**

<i>p</i>	The patcher whose dirty bit will be set.
<i>c</i>	Pass true to lock the patcher's background layer, otherwise pass false.

**Returns**

A Max error code.

**38.41.2.35 jpatcher\_set\_defrect()**

```
t_max_err jpatcher_set_defrect (
    t_object * p,
    t_rect * pr )
```

Set a patcher's default location and size.

**Parameters**

<i>p</i>	A pointer to a patcher instance.
<i>pr</i>	The address of a <a href="#">t_rect</a> with the new position and size.

**Returns**

A Max error code.

**38.41.2.36 jpatcher\_set\_dirty()**

```
t_max_err jpatcher_set_dirty (
    t_object * p,
    char c )
```

Set a patcher's dirty bit.

**Parameters**

<i>p</i>	The patcher whose dirty bit will be set.
<i>c</i>	The new value for the patcher's dirty bit (pass true or false).

**Returns**

A Max error code.

**38.41.2.37 jpatcher\_set\_editing\_bgcolor()**

```
t_max_err jpatcher_set_editing_bgcolor (
    t_object * p,
    t_jrgba * prgba )
```

Set a patcher's editing background color.

**Parameters**

<i>p</i>	The patcher to be queried.
<i>prgba</i>	The address of a <a href="#">t_jrgba</a> struct containing the new color to use.

**Returns**

A Max error code.

**38.41.2.38 jpatcher\_set\_fghidden()**

```
t_max_err jpatcher_set_fghidden (
    t_object * p,
    char c )
```

Set whether a patcher's foreground layer is hidden.

**Parameters**

<i>p</i>	The patcher whose dirty bit will be set.
<i>c</i>	Pass true to hide the patcher's foreground layer, otherwise pass false.

**Returns**

A Max error code.

**38.41.2.39 jpatcher\_set\_gridsize()**

```
t_max_err jpatcher_set_gridsize (
    t_object * p,
    double gridsizeX,
    double gridsizeY )
```

Set a patcher's grid size.

**Parameters**

<i>p</i>	The patcher to be queried.
<i>gridsizeX</i>	The new horizontal grid spacing for the patcher.
<i>gridsizeY</i>	The new vertical grid spacing for the patcher.

**Returns**

A Max error code.

**38.41.2.40 jpatcher\_set\_locked()**

```
t_max_err jpatcher_set_locked (
    t_object * p,
    char c )
```

Lock or unlock a patcher.

**Parameters**

<i>p</i>	The patcher whose locked state will be changed.
<i>c</i>	Pass true to lock a patcher, otherwise pass false.

**Returns**

A Max error code.

**38.41.2.41 jpatcher\_set\_locked\_bgcolor()**

```
t_max_err jpatcher_set_locked_bgcolor (
    t_object * p,
    t_jrgba * prgba )
```

Set a patcher's locked background color.

**Parameters**

<i>p</i>	The patcher to be queried.
<i>prgba</i>	The address of a <a href="#">t_jrgba</a> struct containing the new color to use.

**Returns**

A Max error code.

### 38.41.2.42 jpatcher\_set\_presentation()

```
t_max_err jpatcher_set_presentation (
    t_object * p,
    char c )
```

Set a patcher to presentation mode.

#### Parameters

<i>p</i>	The patcher whose locked state will be changed.
<i>c</i>	Pass true to switch the patcher to presentation mode, otherwise pass false.

#### Returns

A Max error code.

### 38.41.2.43 jpatcher\_set\_rect()

```
t_max_err jpatcher_set_rect (
    t_object * p,
    t_rect * pr )
```

Set a patcher's location and size.

#### Parameters

<i>p</i>	A pointer to a patcher instance.
<i>pr</i>	The address of a <code>t_rect</code> with the new position and size.

#### Returns

A Max error code.

### 38.41.2.44 jpatcher\_set\_title()

```
t_max_err jpatcher_set_title (
    t_object * p,
    t_symbol * ps )
```

Set a patcher's title.

**Parameters**

<i>p</i>	The patcher whose locked state will be changed.
<i>ps</i>	The new title for the patcher.

**Returns**

A Max error code.

**38.41.2.45 jpatcher\_uniqueboxname()**

```
t_symbol* jpatcher_uniqueboxname (
    t_object * p,
    t_symbol * classname )
```

Generate a unique name for a box in patcher.

**Parameters**

<i>p</i>	A pointer to a patcher instance.
<i>classname</i>	The name of an object's class.

**Returns**

The newly-generated unique name.

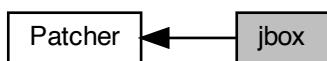
**Remarks**

This is the function used by pattr to assign names to objects in a patcher.

**38.42 jbox**

A box in the patcher.

Collaboration diagram for jbox:



## Data Structures

- struct [t\\_jboxdrawparams](#)  
*The t\_jboxdrawparams structure.*

## Macros

- [#define JBOX\\_DRAWFIRSTIN](#)  
*draw first inlet*
- [#define JBOX\\_NODRAWBOX](#)  
*don't draw the frame*
- [#define JBOX\\_DRAWINLAST](#)  
*draw inlets after update method*
- [#define JBOX\\_TRANSPARENT](#)  
*don't make transparent unless you need it (for efficiency)*
- [#define JBOX\\_NOGROW](#)  
*don't even draw grow thingie*
- [#define JBOX\\_GROWY](#)  
*can grow in y direction by dragging*
- [#define JBOX\\_GROWBOTH](#)  
*can grow independently in both x and y*
- [#define JBOX\\_IGNORELOCKCLICK](#)  
*box should ignore a click if patcher is locked*
- [#define JBOX\\_HILITE](#)  
*flag passed to `jbox_new()` to tell max that the UI object can receive the focus when clicked on – may be replaced by JBOX\_FOCUS in the future*
- [#define JBOX\\_BACKGROUND](#)  
*immediately set box into the background*
- [#define JBOX\\_NOFLOATINSPECTOR](#)  
*no floating inspector window*
- [#define JBOX\\_TEXTFIELD](#)  
*save/load text from textfield, unless JBOX\_BINBUF flag is set*
- [#define JBOX\\_FIXWIDTH](#)  
*give the box a textfield based fix-width (bfixwidth) method*
- [#define JBOX\\_FONTATTR](#)  
*if you want font related attribute you must add this to jbox\_initclass()*
- [#define JBOX\\_TEXTJUSTIFICATIONATTR](#)  
*give your object a textjustification attr to control textfield*
- [#define JBOX\\_BINBUF](#)  
*save/load text from b\_binbuf*
- [#define JBOX\\_MOUSEDRAGDELTA](#)  
*hides mouse cursor in drag and sends mousedragdelta instead of mousdrag (for infinite scrolling like number)*
- [#define JBOX\\_COLOR](#)  
*support the "color" method for color customization*
- [#define JBOX\\_DRAWIOLOCKED](#)  
*draw inlets and outlets when locked (default is not to draw them)*
- [#define JBOX\\_DRAWBACKGROUND](#)

- `#define JBOX_NOINSPECTFIRSTIN`  
*set to have box bg filled in for you based on getdrawparams method or brgba attribute*
- `#define JBOX_FOCUS`  
*flag for objects such as bpatcher that have a different b\_firstin,*  
*more advanced focus support (passed to jbox\_initclass() to add "nextfocus" and "prevfocus" attributes to the UI object).*  
*Not implemented as of 2009-05-11*
- `#define JBOX_BOXVIEW`  
*enable jboxview methods*
- `#define JBOX_LEGACYCOLOR`  
*add undocumented color N message to objects from Max 4 that used it*
- `#define JBOX_COPYLEGACYDEFAULT`  
*if there is a legacy default, copy it instead of the regular default*
- `#define JBOX_NOLEGACYDEFAULT`  
*if there is a legacy default, don't copy any default*

## Enumerations

- enum { `JBOX_FONTFACE_REGULAR` , `JBOX_FONTFACE_BOLD` , `JBOX_FONTFACE_ITALIC` , `JBOX_FONTFACE_BOLDITALIC` }  
*actual numerical values of the b\_fontface attribute; use jbox\_fontface() to weight*
- enum `HitTestResult` {  
`HitNothing` , `HitBox` , `HitInlet` , `HitOutlet` ,  
`HitGrowBox` , `HitLine` , `HitLineLocked` , `HitBorder` }  
*enumerations used for box decorators*

## Functions

- `t_max_err jbox_get_rect_for_view (t_object *box, t_object *patcherview, t_rect *rect)`  
*Find the rect for a box in a given patcherview.*
- `t_max_err jbox_set_rect_for_view (t_object *box, t_object *patcherview, t_rect *rect)`  
*Change the rect for a box in a given patcherview.*
- `t_max_err jbox_get_rect_for_sym (t_object *box, t_symbol *which, t_rect *pr)`  
*Find the rect for a box with a given attribute name.*
- `t_max_err jbox_set_rect_for_sym (t_object *box, t_symbol *which, t_rect *pr)`  
*Change the rect for a box with a given attribute name.*
- `t_max_err jbox_set_rect (t_object *box, t_rect *pr)`  
*Set both the presentation rect and the patching rect.*
- `t_max_err jbox_get_patching_rect (t_object *box, t_rect *pr)`  
*Retrieve the patching rect of a box.*
- `t_max_err jbox_set_patching_rect (t_object *box, t_rect *pr)`  
*Change the patching rect of a box.*
- `t_max_err jbox_get_presentation_rect (t_object *box, t_rect *pr)`  
*Retrieve the presentation rect of a box.*
- `t_max_err jbox_set_presentation_rect (t_object *box, t_rect *pr)`  
*Change the presentation rect of a box.*
- `t_max_err jbox_set_position (t_object *box, t_pt *pos)`

- `t_max_err jbox_get_patching_position (t_object *box, t_pt *pos)`

*Set the position of a box for both the presentation and patching views.*
- `t_max_err jbox_set_patching_position (t_object *box, t_pt *pos)`

*Fetch the position of a box for the patching view.*
- `t_max_err jbox_get_presentation_position (t_object *box, t_pt *pos)`

*Set the position of a box for the patching view.*
- `t_max_err jbox_set_presentation_position (t_object *box, t_pt *pos)`

*Fetch the position of a box for the presentation view.*
- `t_max_err jbox_set_size (t_object *box, t_size *size)`

*Set the size of a box for both the presentation and patching views.*
- `t_max_err jbox_get_patching_size (t_object *box, t_size *size)`

*Fetch the size of a box for the patching view.*
- `t_max_err jbox_set_patching_size (t_object *box, t_size *size)`

*Set the size of a box for the patching view.*
- `t_max_err jbox_get_presentation_size (t_object *box, t_size *size)`

*Fetch the size of a box for the presentation view.*
- `t_max_err jbox_set_presentation_size (t_object *box, t_size *size)`

*Set the size of a box for the presentation view.*
- `t_symbol * jbox_get_maxclass (t_object *b)`

*Retrieve the name of the class of the box's object.*
- `t_object * jbox_get_object (t_object *b)`

*Retrieve a pointer to the box's object.*
- `t_object * jbox_get_patcher (t_object *b)`

*Retrieve a box's patcher.*
- `char jbox_get_hidden (t_object *b)`

*Retrieve a box's 'hidden' attribute.*
- `t_max_err jbox_set_hidden (t_object *b, char c)`

*Set a box's 'hidden' attribute.*
- `t_symbol * jbox_get_fontname (t_object *b)`

*Retrieve a box's 'fontname' attribute.*
- `t_max_err jbox_set_fontname (t_object *b, t_symbol *ps)`

*Set a box's 'fontname' attribute.*
- `double jbox_get_fontsize (t_object *b)`

*Retrieve a box's 'fontsize' attribute.*
- `t_max_err jbox_set_fontsize (t_object *b, double d)`

*Set a box's 'fontsize' attribute.*
- `t_max_err jbox_get_color (t_object *b, t_jrgba *prgba)`

*Retrieve a box's 'color' attribute.*
- `t_max_err jbox_set_color (t_object *b, t_jrgba *prgba)`

*Set a box's 'color' attribute.*
- `t_symbol * jbox_get_hint (t_object *b)`

*Retrieve a box's hint text as a symbol.*
- `t_max_err jbox_set_hint (t_object *b, t_symbol *s)`

*Set a box's hint text using a symbol.*
- `char * jbox_get_hintstring (t_object *bb)`

*Retrieve a box's hint text as a C-string.*

- void `jbox_set_hintstring (t_object *bb, char *s)`  
*Set a box's hint text using a C-string.*
- char \* `jbox_get_annotation (t_object *bb)`  
*Retrieve a box's annotation string, if the user has given it an annotation.*
- void `jbox_set_annotation (t_object *bb, char *s)`  
*Set a box's annotation string.*
- t\_object \* `jbox_get_nextobject (t_object *b)`  
*The next box in the patcher's (linked) list of boxes.*
- t\_object \* `jbox_get_prevobject (t_object *b)`  
*The previous box in the patcher's (linked) list of boxes.*
- t\_symbol \* `jbox_get_varname (t_object *b)`  
*Retrieve a box's scripting name.*
- t\_max\_err `jbox_set_varname (t_object *b, t_symbol *ps)`  
*Set a box's scripting name.*
- t\_symbol \* `jbox_get_id (t_object *b)`  
*Retrieve a boxes unique id.*
- char `jbox_get_canhilite (t_object *b)`  
*Retrieve a box flag value from a box.*
- char `jbox_get_background (t_object *b)`  
*Determine whether a box is located in the patcher's background layer.*
- t\_max\_err `jbox_set_background (t_object *b, char c)`  
*Set whether a box should be in the background or foreground layer of a patcher.*
- char `jbox_get_ignoreclick (t_object *b)`  
*Determine whether a box ignores clicks.*
- t\_max\_err `jbox_set_ignoreclick (t_object *b, char c)`  
*Set whether a box ignores clicks.*
- char `jbox_get_drawfirstin (t_object *b)`  
*Determine whether a box draws its first inlet.*
- char `jbox_get_outline (t_object *b)`  
*Determine whether a box draws an outline.*
- t\_max\_err `jbox_set_outline (t_object *b, char c)`  
*Set whether a box draws an outline.*
- char `jbox_get_growy (t_object *b)`  
*Retrieve a box flag value from a box.*
- char `jbox_get_growboth (t_object *b)`  
*Retrieve a box flag value from a box.*
- char `jbox_get_nogrow (t_object *b)`  
*Retrieve a box flag value from a box.*
- char `jbox_get_drawinlast (t_object *b)`  
*Retrieve a box flag value from a box.*
- t\_object \* `jbox_get_textfield (t_object *b)`  
*Retrieve a pointer to a box's textfield.*
- char `jbox_get_presentation (t_object *b)`  
*Determine if a box is included in the presentation view.*
- t\_max\_err `jbox_set_presentation (t_object *b, char c)`  
*Determine if a box is included in the presentation view.*
- t\_max\_err `jbox_new (t_box *b, long flags, long argc, t_atom *argv)`

- void `jbox_free (t_jbox *b)`  
*Set up your UI object's `t_jbox` member.*
- void `jbox_ready (t_jbox *b)`  
*Tear down your UI object's `t_jbox` member.*
- void `jbox_redraw (t_jbox *b)`  
*Mark the box ready to be accessed and drawn by Max.*
- void `jbox_notify (t_jbox *b, t_symbol *s, t_symbol *msg, void *sender, void *data)`  
*Request that your object/box be re-drawn by Max.*
- `t_max_err jbox_notify (t_jbox *b, t_symbol *s, t_symbol *msg, void *sender, void *data)`  
*Send a notification to a box.*

### 38.42.1 Detailed Description

A box in the patcher.

### 38.42.2 Macro Definition Documentation

#### 38.42.2.1 JBOX\_NOINSPECTFIRSTIN

```
#define JBOX_NOINSPECTFIRSTIN
```

flag for objects such as bpatcher that have a different `b_firstin`,  
 but the attrs of the `b_firstin` should not be shown in the inspector

### 38.42.3 Enumeration Type Documentation

#### 38.42.3.1 anonymous enum

```
anonymous enum
```

actual numerical values of the `b_fontface` attribute; use `jbox_fontface()` to weight

##### Enumerator

<code>JBOX_FONTFACE_REGULAR</code>	normal
<code>JBOX_FONTFACE_BOLD</code>	bold
<code>JBOX_FONTFACE_ITALIC</code>	italic
<code>JBOX_FONTFACE_BOLDITALIC</code>	bold and italic

### 38.42.3.2 HitTestResult

enum `HitTestResult`

enumerations used for box decorators

Enumerator

<code>HitNothing</code>	a hole in the box
<code>HitBox</code>	the body of the box
<code>HitInlet</code>	an inlet
<code>HitOutlet</code>	an outlet
<code>HitGrowBox</code>	the grow handle
<code>HitLine</code>	a line
<code>HitLineLocked</code>	a line in a locked patcher (for probing)
<code>HitBorder</code>	border around the box (drawn when selected), can use to select or move (only used when patch is unlocked)

### 38.42.4 Function Documentation

#### 38.42.4.1 `jbox_free()`

```
void jbox_free (
    t_jbox * b )
```

Tear down your UI object's `t_jbox` member.

This should be called from your UI object's free method.

Parameters

<code>b</code>	The address of your object's <code>t_jbox</code> member (which should be the first member of the object's struct).
----------------	--

#### 38.42.4.2 `jbox_get_annotation()`

```
char* jbox_get_annotation (
    t_object * bb )
```

Retrieve a box's annotation string, if the user has given it an annotation.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The user-created annotation string for a box, or NULL if no string exists.

**38.42.4.3 *jbox\_get\_background()***

```
char jbox_get_background (
    t_object * b )
```

Determine whether a box is located in the patcher's background layer.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

Zero if the object is in the foreground, otherwise non-zero.

**38.42.4.4 *jbox\_get\_canhilite()***

```
char jbox_get_canhilite (
    t_object * b )
```

Retrieve a box flag value from a box.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The value of the canhilite bit in the box's flags.

### 38.42.4.5 `jbox_get_color()`

```
t_max_err jbox_get_color (
    t_object * b,
    t_jrgba * prgba )
```

Retrieve a box's 'color' attribute.

#### Parameters

<code>b</code>	The box to query.
<code>prgba</code>	The address of a valid <code>t_rect</code> whose values will be filled-in upon return.

#### Returns

A Max error code.

### 38.42.4.6 `jbox_get_drawfirstin()`

```
char jbox_get_drawfirstin (
    t_object * b )
```

Determine whether a box draws its first inlet.

#### Parameters

<code>b</code>	The box to query.
----------------	-------------------

#### Returns

Zero if the inlet is not drawn, otherwise non-zero.

### 38.42.4.7 `jbox_get_drawinlast()`

```
char jbox_get_drawinlast (
    t_object * b )
```

Retrieve a box flag value from a box.

#### Parameters

<code>b</code>	The box to query.
----------------	-------------------

**Returns**

The value of the drawinlast bit in the box's flags.

**38.42.4.8 jbox\_get\_fontname()**

```
t_symbol* jbox_get_fontname (
    t_object * b )
```

Retrieve a box's 'fontname' attribute.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The font name.

**38.42.4.9 jbox\_get\_fontsize()**

```
double jbox_get_fontsize (
    t_object * b )
```

Retrieve a box's 'fontsize' attribute.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The font size in points.

**38.42.4.10 jbox\_get\_growboth()**

```
char jbox_get_growboth (
    t_object * b )
```

Retrieve a box flag value from a box.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The value of the growboth bit in the box's flags.

**38.42.4.11 jbox\_get\_growy()**

```
char jbox_get_growy (
    t_object * b )
```

Retrieve a box flag value from a box.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The value of the growy bit in the box's flags.

**38.42.4.12 jbox\_get\_hidden()**

```
char jbox_get_hidden (
    t_object * b )
```

Retrieve a box's 'hidden' attribute.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

True if the box is hidden, otherwise false.

**38.42.4.13 jbox\_get\_hint()**

```
t_symbol* jbox_get_hint (
    t_object * b )
```

Retrieve a box's hint text as a symbol.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The box's hint text.

**38.42.4.14 jbox\_get\_hintstring()**

```
char* jbox_get_hintstring (
    t_object * bb )
```

Retrieve a box's hint text as a C-string.

**Parameters**

<i>bb</i>	The box to query.
-----------	-------------------

**Returns**

The box's hint text.

**38.42.4.15 jbox\_get\_id()**

```
t_symbol* jbox_get_id (
    t_object * b )
```

Retrieve a boxes unique id.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The unique id of the object. This is a symbol that is referenced, for example, by patchlines.

**38.42.4.16 jbox\_get\_ignoreclick()**

```
char jbox_get_ignoreclick (
    t_object * b )
```

Determine whether a box ignores clicks.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

Zero if the object responds to clicks, otherwise non-zero.

**38.42.4.17 jbox\_get\_maxclass()**

```
t_symbol* jbox_get_maxclass (
    t_object * b )
```

Retrieve the name of the class of the box's object.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The name of the class of the box's object.

**38.42.4.18 jbox\_get\_nextobject()**

```
t_object* jbox_get_nextobject (
    t_object * b )
```

The next box in the patcher's (linked) list of boxes.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The next box in the list.

**38.42.4.19 jbox\_get\_nogrow()**

```
char jbox_get_nogrow (
    t_object * b )
```

Retrieve a box flag value from a box.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The value of the nogrow bit in the box's flags.

**38.42.4.20 jbox\_get\_object()**

```
t_object* jbox_get_object (
    t_object * b )
```

Retrieve a pointer to the box's object.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

A pointer to the box's object.

### 38.42.4.21 jbox\_get\_outline()

```
char jbox_get_outline (
    t_object * b )
```

Determine whether a box draws an outline.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

Zero if the outline is not drawn, otherwise non-zero.

### 38.42.4.22 jbox\_get\_patcher()

```
t_object* jbox_get_patcher (
    t_object * b )
```

Retrieve a box's patcher.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

If the box has a patcher, the patcher's pointer is returned. Otherwise NULL is returned.

### 38.42.4.23 jbox\_get\_patching\_position()

```
t_max_err jbox_get_patching_position (
    t_object * box,
    t_pt * pos )
```

Fetch the position of a box for the patching view.

**Parameters**

<i>box</i>	The box whose position will be retrieved.
<i>pos</i>	The address of a valid <i>t_pt</i> whose x and y values will be filled in.

**Returns**

A Max error code.

**38.42.4.24 jbox\_get\_patching\_rect()**

```
t_max_err jbox_get_patching_rect (
    t_object * box,
    t_rect * pr )
```

Retrieve the patching rect of a box.

**Parameters**

<i>box</i>	The box whose rect values will be retrieved.
<i>pr</i>	The address of a valid <code>t_rect</code> whose values will be filled in.

**Returns**

A Max error code.

**38.42.4.25 jbox\_get\_patching\_size()**

```
t_max_err jbox_get_patching_size (
    t_object * box,
    t_size * size )
```

Fetch the size of a box for the patching view.

**Parameters**

<i>box</i>	The box whose size will be retrieved.
<i>size</i>	The address of a valid <code>t_size</code> whose width and height values will be filled in.

**Returns**

A Max error code.

### 38.42.4.26 jbox\_get\_presentation()

```
char jbox_get_presentation (
    t_object * b )
```

Determine if a box is included in the presentation view.

#### Parameters

<i>b</i>	The box to query.
----------	-------------------

#### Returns

Non-zero if in presentation mode, otherwise zero.

### 38.42.4.27 jbox\_get\_presentation\_position()

```
t_max_err jbox_get_presentation_position (
    t_object * box,
    t_pt * pos )
```

Fetch the position of a box for the presentation view.

#### Parameters

<i>box</i>	The box whose position will be retrieved.
<i>pos</i>	The address of a valid <a href="#">t_pt</a> whose x and y values will be filled in.

#### Returns

A Max error code.

### 38.42.4.28 jbox\_get\_presentation\_rect()

```
t_max_err jbox_get_presentation_rect (
    t_object * box,
    t_rect * pr )
```

Retrieve the presentation rect of a box.

**Parameters**

<i>box</i>	The box whose rect values will be retrieved.
<i>pr</i>	The address of a valid <a href="#">t_rect</a> whose values will be filled in.

**Returns**

A Max error code.

**38.42.4.29 jbox\_get\_presentation\_size()**

```
t_max_err jbox_get_presentation_size (
    t_object * box,
    t_size * size )
```

Fetch the size of a box for the presentation view.

**Parameters**

<i>box</i>	The box whose size will be retrieved.
<i>size</i>	The address of a valid <a href="#">t_size</a> whose width and height values will be filled in.

**Returns**

A Max error code.

**38.42.4.30 jbox\_get\_prevobject()**

```
t_object* jbox_get_prevobject (
    t_object * b )
```

The previous box in the patcher's (linked) list of boxes.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The next box in the list.

### 38.42.4.31 jbox\_get\_rect\_for\_sym()

```
t_max_err jbox_get_rect_for_sym (
    t_object * box,
    t_symbol * which,
    t_rect * pr )
```

Find the rect for a box with a given attribute name.

#### Parameters

<i>box</i>	The box whose rect will be fetched.
<i>which</i>	The name of the rect attribute to be fetched, for example <code>_sym_presentation_rect</code> or <code>_sym_patchning_rect</code> .
<i>pr</i>	The address of a valid <code>t_rect</code> whose members will be filled in by this function.

#### Returns

A Max error code.

### 38.42.4.32 jbox\_get\_rect\_for\_view()

```
t_max_err jbox_get_rect_for_view (
    t_object * box,
    t_object * patcherview,
    t_rect * rect )
```

Find the rect for a box in a given patcherview.

#### Parameters

<i>box</i>	The box whose rect will be fetched.
<i>patcherview</i>	A patcherview in which the box exists.
<i>rect</i>	The address of a valid <code>t_rect</code> whose members will be filled in by this function.

#### Returns

A Max error code.

**38.42.4.33 jbox\_get\_textfield()**

```
t_object* jbox_get_textfield (
    t_object * b )
```

Retrieve a pointer to a box's textfield.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The textfield for the box, assuming it has one. If the box does not own a textfield then NULL is returned.

**38.42.4.34 jbox\_get\_varname()**

```
t_symbol* jbox_get_varname (
    t_object * b )
```

Retrieve a box's scripting name.

**Parameters**

<i>b</i>	The box to query.
----------	-------------------

**Returns**

The box's scripting name.

**38.42.4.35 jbox\_new()**

```
t_max_err jbox_new (
    t_jbox * b,
    long flags,
    long argc,
    t_atom * argv )
```

Set up your UI object's `t_jbox` member.

This should be called from your UI object's free method.

**Parameters**

<i>b</i>	The address of your UI object's <code>t_jbox</code> member (which should be the first member of the object's struct).
<i>flags</i>	Flags to set the box's behavior, such as <code>JBOX_NODRAWBOX</code> .
<i>argc</i>	The count of atoms in the <code>argv</code> parameter.
<i>argv</i>	The address of the first in an array of atoms to be passed to the box constructor. Typically these are simply the argument passed to your object when it is created.

**Returns**

A Max error code.

**38.42.4.36 jbox\_notify()**

```
t_max_err jbox_notify (
    t_jbox * b,
    t_symbol * s,
    t_symbol * msg,
    void * sender,
    void * data )
```

Send a notification to a box.

This is the same as calling `object_notify()` for a box.

**Parameters**

<i>b</i>	The address of your object's <code>t_jbox</code> member.
<i>s</i>	The name of the send object.
<i>msg</i>	The notification name.
<i>sender</i>	The sending object's address.
<i>data</i>	A pointer to some data passed to the box's notify method.

**Returns**

A Max error code.

**38.42.4.37 jbox\_ready()**

```
void jbox_ready (
    t_jbox * b )
```

Mark the box ready to be accessed and drawn by Max.

This should typically be called at the end of your UI object's new method.

**Parameters**

<i>b</i>	The address of your object's <a href="#">t_jbox</a> member.
----------	---

**38.42.4.38 jbox\_redraw()**

```
void jbox_redraw (
    t_jbox * b )
```

Request that your object/box be re-drawn by Max.

**Parameters**

<i>b</i>	The address of your object's <a href="#">t_jbox</a> member.
----------	---

**38.42.4.39 jbox\_set\_annotation()**

```
void jbox_set_annotation (
    t_object * bb,
    char * s )
```

Set a box's annotation string.

**Parameters**

<i>bb</i>	The box to query.
<i>s</i>	The annotation string for the box.

**Returns**

A Max error code.

**38.42.4.40 jbox\_set\_background()**

```
t_max_err jbox_set_background (
    t_object * b,
    char c )
```

Set whether a box should be in the background or foreground layer of a patcher.

**Parameters**

<i>b</i>	The box to query.
<i>c</i>	Pass zero to tell the box to appear in the foreground, or non-zero to indicate that the box should be in the background layer.

**Returns**

A Max error code.

**38.42.4.41 jbox\_set\_color()**

```
t_max_err jbox_set_color (
    t_object * b,
    t_jrgba * prgba )
```

Set a box's 'color' attribute.

**Parameters**

<i>b</i>	The box to query.
<i>prgba</i>	The address of a <a href="#">t_rect</a> containing the desired color for the box/object.

**Returns**

A Max error code.

**38.42.4.42 jbox\_set\_fontname()**

```
t_max_err jbox_set_fontname (
    t_object * b,
    t_symbol * ps )
```

Set a box's 'fontname' attribute.

**Parameters**

<i>b</i>	The box to query.
<i>ps</i>	The font name. Note that the font name may be case-sensitive.

**Returns**

A Max error code.

**38.42.4.43 jbox\_set\_fontsize()**

```
t_max_err jbox_set_fontsize (
    t_object * b,
    double d )
```

Set a box's 'fontsize' attribute.

**Parameters**

<i>b</i>	The box to query.
<i>d</i>	The fontsize in points.

**Returns**

A Max error code.

**38.42.4.44 jbox\_set\_hidden()**

```
t_max_err jbox_set_hidden (
    t_object * b,
    char c )
```

Set a box's 'hidden' attribute.

**Parameters**

<i>b</i>	The box to query.
<i>c</i>	Set to true to hide the box, otherwise false.

**Returns**

A Max error code.

**38.42.4.45 jbox\_set\_hint()**

```
t_max_err jbox_set_hint (
    t_object * b,
    t_symbol * s )
```

Set a box's hint text using a symbol.

**Parameters**

<i>b</i>	The box to query.
<i>s</i>	The new text to use for the box's hint.

**Returns**

A Max error code.

**38.42.4.46 jbox\_set\_hintstring()**

```
void jbox_set_hintstring (
    t_object * bb,
    char * s )
```

Set a box's hint text using a C-string.

**Parameters**

<i>bb</i>	The box to query.
<i>s</i>	The new text to use for the box's hint.

**Returns**

A Max error code.

**38.42.4.47 jbox\_set\_ignoreclick()**

```
t_max_err jbox_set_ignoreclick (
    t_object * b,
    char c )
```

Set whether a box ignores clicks.

**Parameters**

<i>b</i>	The box to query.
<i>c</i>	Pass zero to tell the box to respond to clicks, or non-zero to indicate that the box should ignore clicks.

**Returns**

A Max error code.

**38.42.4.48 jbox\_set\_outline()**

```
t_max_err jbox_set_outline (
    t_object * b,
    char c )
```

Set whether a box draws an outline.

**Parameters**

<i>b</i>	The box to query.
<i>c</i>	Pass zero to hide the outline, or non-zero to indicate that the box should draw the outline.

**Returns**

A Max error code.

**38.42.4.49 jbox\_set\_patching\_position()**

```
t_max_err jbox_set_patching_position (
    t_object * box,
    t_pt * pos )
```

Set the position of a box for the patching view.

**Parameters**

<i>box</i>	The box whose positon will be changed.
<i>pos</i>	The address of a <a href="#">t_pt</a> with the new x and y values.

**Returns**

A Max error code.

**38.42.4.50 jbox\_set\_patching\_rect()**

```
t_max_err jbox_set_patching_rect (
    t_object * box,
    t_rect * pr )
```

Change the patching rect of a box.

**Parameters**

<i>box</i>	The box whose rect will be changed.
<i>pr</i>	The address of a <a href="#">t_rect</a> with the new rect values.

**Returns**

A Max error code.

**38.42.4.51 jbox\_set\_patching\_size()**

```
t_max_err jbox_set_patching_size (
    t_object * box,
    t_size * size )
```

Set the size of a box for the patching view.

**Parameters**

<i>box</i>	The box whose size will be changed.
<i>size</i>	The address of a <a href="#">t_size</a> with the new width and height values.

**Returns**

A Max error code.

### 38.42.4.52 jbox\_set\_position()

```
t_max_err jbox_set_position (
    t_object * box,
    t_pt * pos )
```

Set the position of a box for both the presentation and patching views.

#### Parameters

<i>box</i>	The box whose position will be changed.
<i>pos</i>	The address of a <a href="#">t_pt</a> with the new x and y values.

#### Returns

A Max error code.

### 38.42.4.53 jbox\_set\_presentation()

```
t_max_err jbox_set_presentation (
    t_object * b,
    char c )
```

Determine if a box is included in the presentation view.

#### Parameters

<i>b</i>	The box to query.
<i>c</i>	Pass zero to remove a box from the presentation view, or non-zero to add it to the presentation view.

#### Returns

Non-zero if in presentation mode, otherwise zero.

### 38.42.4.54 jbox\_set\_presentation\_position()

```
t_max_err jbox_set_presentation_position (
    t_object * box,
    t_pt * pos )
```

Set the position of a box for the presentation view.

**Parameters**

<i>box</i>	The box whose rect will be changed.
<i>pos</i>	The address of a <a href="#">t_pt</a> with the new x and y values.

**Returns**

A Max error code.

**38.42.4.55 jbox\_set\_presentation\_rect()**

```
t_max_err jbox_set_presentation_rect (
    t_object * box,
    t_rect * pr )
```

Change the presentation rect of a box.

**Parameters**

<i>box</i>	The box whose rect will be changed.
<i>pr</i>	The address of a <a href="#">t_rect</a> with the new rect values.

**Returns**

A Max error code.

**38.42.4.56 jbox\_set\_presentation\_size()**

```
t_max_err jbox_set_presentation_size (
    t_object * box,
    t_size * size )
```

Set the size of a box for the presentation view.

**Parameters**

<i>box</i>	The box whose size will be changed.
<i>size</i>	The address of a <a href="#">t_size</a> with the new width and height values.

**Returns**

A Max error code.

**38.42.4.57 jbox\_set\_rect()**

```
t_max_err jbox_set_rect (
    t_object * box,
    t_rect * pr )
```

Set both the presentation rect and the patching rect.

**Parameters**

<i>box</i>	The box whose rect will be changed.
<i>pr</i>	The address of a <a href="#">t_rect</a> with the new rect values.

**Returns**

A Max error code.

**38.42.4.58 jbox\_set\_rect\_for\_sym()**

```
t_max_err jbox_set_rect_for_sym (
    t_object * box,
    t_symbol * which,
    t_rect * pr )
```

Change the rect for a box with a given attribute name.

**Parameters**

<i>box</i>	The box whose rect will be changed.
<i>which</i>	The name of the rect attribute to be changed, for example <code>_sym_presentation_rect</code> or <code>_sym_patch_rect</code> .
<i>pr</i>	The address of a valid <a href="#">t_rect</a> that will replace the current values used by the box.

**Returns**

A Max error code.

**38.42.4.59 jbox\_set\_rect\_for\_view()**

```
t_max_err jbox_set_rect_for_view (
    t_object * box,
    t_object * patcherview,
    t_rect * rect )
```

Change the rect for a box in a given patcherview.

**Parameters**

<i>box</i>	The box whose rect will be changed.
<i>patcherview</i>	A patcherview in which the box exists.
<i>rect</i>	The address of a valid <a href="#">t_rect</a> that will replace the current values used by the box in the given view.

**Returns**

A Max error code.

**38.42.4.60 jbox\_set\_size()**

```
t_max_err jbox_set_size (
    t_object * box,
    t_size * size )
```

Set the size of a box for both the presentation and patching views.

**Parameters**

<i>box</i>	The box whose size will be changed.
<i>size</i>	The address of a <a href="#">t_size</a> with the new size values.

**Returns**

A Max error code.

**38.42.4.61 jbox\_set\_varname()**

```
t_max_err jbox_set_varname (
    t_object * b,
    t_symbol * ps )
```

Set a box's scripting name.

### Parameters

<i>b</i>	The box to query.
<i>ps</i>	The new scripting name for the box.

### Returns

A Max error code.

## 38.43 jpatchline

A patch cord.

Collaboration diagram for jpatchline:



### Functions

- `t_max_err jpatchline_get_startpoint (t_object *l, double *x, double *y)`  
*Retrieve a patchline's starting point.*
- `t_max_err jpatchline_get_endpoint (t_object *l, double *x, double *y)`  
*Retrieve a patchline's ending point.*
- `long jpatchline_get_nummidpoints (t_object *l)`  
*Determine the number of midpoints (segments) in a patchline.*
- `t_object * jpatchline_get_box1 (t_object *l)`  
*Return the object box from which a patchline originates.*
- `long jpatchline_get_outletnum (t_object *l)`  
*Return the outlet number of the originating object box from which a patchline begins.*
- `t_object * jpatchline_get_box2 (t_object *l)`  
*Return the destination object box for a patchline.*
- `long jpatchline_get_inletnum (t_object *l)`  
*Return the inlet number of the destination object box to which a patchline is connected.*
- `t_object * jpatchline_get_nextline (t_object *b)`  
*Given a patchline, traverse to the next patchline in the (linked) list.*
- `char jpatchline_get_hidden (t_object *l)`  
*Determine if a patch line is hidden.*
- `t_max_err jpatchline_set_hidden (t_object *l, char c)`  
*Set a patchline's visibility.*
- `t_max_err jpatchline_get_color (t_object *l, t_jrgba *prgba)`  
*Get the color of a patch line.*
- `t_max_err jpatchline_set_color (t_object *l, t_jrgba *prgba)`  
*Set the color of a patch line.*

### 38.43.1 Detailed Description

A patch cord.

### 38.43.2 Function Documentation

#### 38.43.2.1 jpatchline\_get\_box1()

```
t_object* jpatchline_get_box1 (
    t_object * l )
```

Return the object box from which a patchline originates.

##### Parameters

/	A pointer to the patchline's instance.
---	--

##### Returns

The object box from which the patchline originates.

#### 38.43.2.2 jpatchline\_get\_box2()

```
t_object* jpatchline_get_box2 (
    t_object * l )
```

Return the destination object box for a patchline.

##### Parameters

/	A pointer to the patchline's instance.
---	--

##### Returns

The destination object box for a patchline.

### 38.43.2.3 jpatchline\_get\_color()

```
t_max_err jpatchline_get_color (
    t_object * l,
    t_jrgba * prgba )
```

Get the color of a patch line.

#### Parameters

<i>l</i>	A patchline instance.
<i>prgba</i>	The address of a valid <a href="#">t_jrgba</a> struct that will be filled with the color values of the patch line.

#### Returns

An error code.

### 38.43.2.4 jpatchline\_get\_endpoint()

```
t_max_err jpatchline_get_endpoint (
    t_object * l,
    double * x,
    double * y )
```

Retrieve a patchline's ending point.

#### Parameters

<i>l</i>	A pointer to the patchline's instance.
<i>x</i>	The address of a variable to hold the x-coordinate of the ending point's position upon return.
<i>y</i>	The address of a variable to hold the y-coordinate of the ending point's position upon return.

#### Returns

A Max error code.

### 38.43.2.5 jpatchline\_get\_hidden()

```
char jpatchline_get_hidden (
    t_object * l )
```

Determine if a patch line is hidden.

**Parameters**

/	A patchline instance.
---	-----------------------

**Returns**

Zero if the patchline is visible, non-zero if it is hidden.

**38.43.2.6 jpatchline\_get\_inletnum()**

```
long jpatchline_get_inletnum (
    t_object * l )
```

Return the inlet number of the destination object box to which a patchline is connected.

**Parameters**

/	A pointer to the patchline's instance.
---	--

**Returns**

The inlet number.

**38.43.2.7 jpatchline\_get\_nextline()**

```
t_object* jpatchline_get_nextline (
    t_object * b )
```

Given a patchline, traverse to the next patchline in the (linked) list.

**Parameters**

b	A patchline instance.
---	-----------------------

**Returns**

The next patchline. If the current patchline is at the end (tail) of the list, then NULL is returned.

### 38.43.2.8 jpatchline\_get\_nummidpoints()

```
long jpatchline_get_nummidpoints (
    t_object * l )
```

Determine the number of midpoints (segments) in a patchline.

#### Parameters

/	A pointer to the patchline's instance.
---	--

#### Returns

The number of midpoints in the patchline.

### 38.43.2.9 jpatchline\_get\_outletnum()

```
long jpatchline_get_outletnum (
    t_object * l )
```

Return the outlet number of the originating object box from which a patchline begins.

#### Parameters

/	A pointer to the patchline's instance.
---	--

#### Returns

The outlet number.

### 38.43.2.10 jpatchline\_get\_startpoint()

```
t_max_err jpatchline_get_startpoint (
    t_object * l,
    double * x,
    double * y )
```

Retrieve a patchline's starting point.

#### Parameters

/	A pointer to the patchline's instance.
x	The address of a variable to hold the x-coordinate of the starting point's position upon return.
y	The address of a variable to hold the y-coordinate of the starting point's position upon return.

**Returns**

A Max error code.

**38.43.2.11 jpatchline\_set\_color()**

```
t_max_err jpatchline_set_color (
    t_object * l,
    t_jrgba * prgba )
```

Set the color of a patch line.

**Parameters**

<i>l</i>	A patchline instance.
<i>prgba</i>	The address of a valid <code>t_jrgba</code> struct containing the color to use.

**Returns**

An error code.

**38.43.2.12 jpatchline\_set\_hidden()**

```
t_max_err jpatchline_set_hidden (
    t_object * l,
    char c )
```

Set a patchline's visibility.

**Parameters**

<i>l</i>	A patchline instance.
<i>c</i>	Pass 0 to make a patchline visible, or non-zero to hide it.

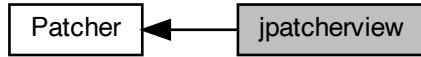
**Returns**

An error code.

**38.44 jpatcherview**

A view of a patcher.

Collaboration diagram for jpatcherview:



## Functions

- `t_object * patcherview_findpatcherview (int x, int y)`  
*Find a patcherview at the given screen coords.*
- `char patcherview_get_visible (t_object *pv)`  
*Query a patcherview to determine whether it is visible.*
- `t_max_err patcherview_set_visible (t_object *pv, char c)`  
*Set the 'visible' attribute of a patcherview.*
- `t_max_err patcherview_get_rect (t_object *pv, t_rect *pr)`  
*Get the value of the rect attribute for a patcherview.*
- `t_max_err patcherview_set_rect (t_object *pv, t_rect *pr)`  
*Set the value of the rect attribute for a patcherview.*
- `void patcherview_canvas_to_screen (t_object *pv, double cx, double cy, long *sx, long *sy)`  
*Convert the point cx, cy in canvas coordinates to screen coordinates.*
- `void patcherview_screen_to_canvas (t_object *pv, long sx, long sy, double *cx, double *cy)`  
*Convert the point cx, cy in screen coordinates to canvas coordinates.*
- `char patcherview_get_locked (t_object *p)`  
*Find out if a patcherview is locked.*
- `t_max_err patcherview_set_locked (t_object *p, char c)`  
*Lock or unlock a patcherview.*
- `char patcherview_get_presentation (t_object *pv)`  
*Find out if a patcherview is a presentation view.*
- `t_max_err patcherview_set_presentation (t_object *p, char c)`  
*Set whether or not a patcherview is a presentation view.*
- `double patcherview_get_zoomfactor (t_object *pv)`  
*Fetch the zoom-factor of a patcherview.*
- `t_max_err patcherview_set_zoomfactor (t_object *pv, double d)`  
*Set the zoom-factor of a patcherview.*
- `t_object * patcherview_get_nextview (t_object *pv)`  
*Given a patcherview, find the next patcherview.*
- `t_object * patcherview_get_jgraphics (t_object *pv)`  
*Given a patcherview, return the `t_jgraphics` context for that view.*
- `t_object * patcherview_get_patcher (t_object *pv)`  
*Given a patcherview, return its patcher.*
- `t_object * patcherview_get_topview (t_object *pv)`  
*Given a patcherview, return the top patcherview (possibly itself).*

### 38.44.1 Detailed Description

A view of a patcher.

### 38.44.2 Function Documentation

#### 38.44.2.1 patcherview\_canvas\_to\_screen()

```
void patcherview_canvas_to_screen (
    t_object * pv,
    double cx,
    double cy,
    long * sx,
    long * sy )
```

Convert the point cx, cy in canvas coordinates to screen coordinates.

##### Parameters

<i>pv</i>	The patcherview instance the canvas coords are relative to.
<i>cx</i>	The x dimension of the canvas coordinate relative to the patcherview.
<i>cy</i>	The y dimension of the canvas coordinate relative to the patcherview.
<i>sx</i>	A pointer to a long to receive the screen coordinate x dimension.
<i>sy</i>	A pointer to a long to receive the screen coordinate y dimension.

#### 38.44.2.2 patcherview\_findpatcherview()

```
t_object* patcherview_findpatcherview (
    int x,
    int y )
```

Find a patcherview at the given screen coords.

##### Parameters

<i>x</i>	The horizontal coordinate at which to find a patcherview.
<i>y</i>	The vertical coordinate at which to find a patcherview.

**Returns**

A pointer to the patcherview at the specified location, or NULL if no patcherview exists at that location.

**38.44.2.3 patcherview\_get\_jgraphics()**

```
t_object* patcherview_get_jgraphics (
    t_object * pv )
```

Given a patcherview, return the [t\\_jgraphics](#) context for that view.

**Parameters**

<i>pv</i>	The patcherview instance.
-----------	---------------------------

**Returns**

The [t\\_jgraphics](#) context for the view.

**38.44.2.4 patcherview\_get\_locked()**

```
char patcherview_get_locked (
    t_object * p )
```

Find out if a patcherview is locked.

**Parameters**

<i>p</i>	The patcherview instance whose attribute value will be fetched.
----------	---

**Returns**

Returns 0 if unlocked, otherwise returns non-zero.

**38.44.2.5 patcherview\_get\_nextview()**

```
t_object* patcherview_get_nextview (
    t_object * pv )
```

Given a patcherview, find the next patcherview.

The views of a patcher are maintained internally as a [t\\_linklist](#), and so the views can be traversed should you need to perform operations on all of a patcher's patcherviews.

**Parameters**

<i>pv</i>	The patcherview instance from which to find the next patcherview.
-----------	---

**Returns**

The next patcherview in the list, or NULL if the patcherview passed in *pv* is the tail.

**38.44.2.6 patcherview\_get\_patcher()**

```
t_object* patcherview_get_patcher (
    t_object * pv )
```

Given a patcherview, return its patcher.

**Parameters**

<i>pv</i>	The patcherview instance for which to fetch the patcher.
-----------	--

**Returns**

The patcher.

**38.44.2.7 patcherview\_get\_presentation()**

```
char patcherview_get_presentation (
    t_object * pv )
```

Find out if a patcherview is a presentation view.

**Parameters**

<i>pv</i>	The patcherview instance whose attribute value will be fetched.
-----------	---

**Returns**

Returns 0 if the view is not a presentation view, otherwise returns non-zero.

### 38.44.2.8 patcherview\_get\_rect()

```
t_max_err patcherview_get_rect (
    t_object * pv,
    t_rect * pr )
```

Get the value of the rect attribute for a patcherview.

#### Parameters

<i>pv</i>	The patcherview instance whose attribute value will be fetched.
<i>pr</i>	The address of a valid <a href="#">t_rect</a> struct, whose contents will be filled upon return.

#### Returns

An error code.

### 38.44.2.9 patcherview\_get\_topview()

```
t_object* patcherview_get_topview (
    t_object * pv )
```

Given a patcherview, return the top patcherview (possibly itself).

If the patcherview is inside a bpatcher which is in a patcher then this will give you the view the bpatcher view is inside of.

#### Parameters

<i>pv</i>	The patcherview instance whose top view you want to get.
-----------	--

#### Returns

The top patcherview.

### 38.44.2.10 patcherview\_get\_visible()

```
char patcherview_get_visible (
    t_object * pv )
```

Query a patcherview to determine whether it is visible.

**Parameters**

<i>pv</i>	The patcherview instance to query.
-----------	------------------------------------

**Returns**

Returns zero if the patcherview is invisible, otherwise returns non-zero.

**38.44.2.11 patcherview\_get\_zoomfactor()**

```
double patcherview_get_zoomfactor (
    t_object * pv )
```

Fetch the zoom-factor of a patcherview.

**Parameters**

<i>pv</i>	The patcherview instance whose attribute value will be fetched.
-----------	---

**Returns**

The factor by which the view is zoomed.

**38.44.2.12 patcherview\_screen\_to\_canvas()**

```
void patcherview_screen_to_canvas (
    t_object * pv,
    long sx,
    long sy,
    double * cx,
    double * cy )
```

Convert the point cx, cy in canvas coordinates to screen coordinates.

**Parameters**

<i>pv</i>	The patcherview instance the canvas coords are relative to.
<i>sx</i>	The screen position x coordinate.
<i>sy</i>	The screen position y coordinate
<i>cx</i>	A pointer to a double to receive the canvas coordinate for the given screen x position.
<i>cy</i>	A pointer to a double to receive the canvas coordinate for the given screen y position.

### 38.44.2.13 patcherview\_set\_locked()

```
t_max_err patcherview_set_locked (
    t_object * p,
    char c )
```

Lock or unlock a patcherview.

#### Parameters

<i>p</i>	The patcherview instance whose attribute value will be set.
<i>c</i>	Set this value to zero to unlock the patcherview, otherwise pass a non-zero value.

#### Returns

An error code.

### 38.44.2.14 patcherview\_set\_presentation()

```
t_max_err patcherview_set_presentation (
    t_object * p,
    char c )
```

Set whether or not a patcherview is a presentation view.

#### Parameters

<i>p</i>	The patcherview instance whose attribute value will be set.
<i>c</i>	Set this value to non-zero to make the patcherview a presentation view, otherwise pass zero.

#### Returns

An error code.

### 38.44.2.15 patcherview\_set\_rect()

```
t_max_err patcherview_set_rect (
    t_object * pv,
    t_rect * pr )
```

Set the value of the rect attribute for a patcherview.

**Parameters**

<i>pv</i>	The patcherview instance whose attribute value will be set.
<i>pr</i>	The address of a valid <a href="#">t_rect</a> struct.

**Returns**

An error code.

**38.44.2.16 patcherview\_set\_visible()**

```
t_max_err patcherview_set_visible (
    t_object * pv,
    char c )
```

Set the 'visible' attribute of a patcherview.

**Parameters**

<i>pv</i>	The patcherview instance whose attribute will be set.
<i>c</i>	Whether or not the patcherview should be made visible.

**Returns**

An error code.

**38.44.2.17 patcherview\_set\_zoomfactor()**

```
t_max_err patcherview_set_zoomfactor (
    t_object * pv,
    double d )
```

Set the zoom-factor of a patcherview.

**Parameters**

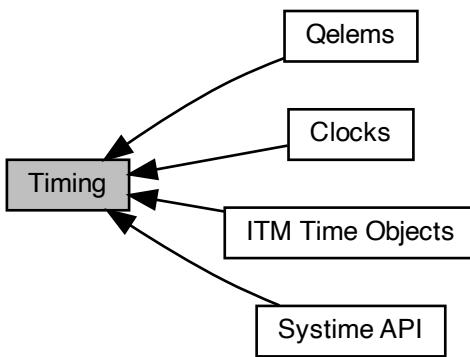
<i>pv</i>	The patcherview instance whose attribute value will be set.
<i>d</i>	The zoom-factor at which the patcherview should display the patcher.

**Returns**

An error code.

## 38.45 Timing

Collaboration diagram for Timing:



## Modules

- [Clocks](#)

*Clock objects are your interface to Max's scheduler.*

- [Qelems](#)

*Your object's methods may be called at interrupt level.*

- [Systime API](#)

*The Systime API provides the means of getting the system time, instead of the scheduler time as you would with functions like `gettime()`.*

- [ITM Time Objects](#)

*ITM Time Objects are a high-level interface to ITM, a tempo-based scheduler API.*

### 38.45.1 Detailed Description

## 38.46 Clocks

Clock objects are your interface to Max's scheduler.

Collaboration diagram for Clocks:



## Functions

- `t_clock * clock_new (void *obj, method fn)`  
*Create a new Clock object.*
- `void clock_delay (t_clock *x, long n)`  
*Schedule the execution of a Clock.*
- `void clock_unset (t_clock *x)`  
*Cancel the scheduled execution of a Clock.*
- `void clock_fset (t_clock *x, double when)`  
*Schedule the execution of a Clock using a floating-point argument.*
- `void clock_gettime (double *time)`  
*Find out the current logical time of the scheduler in milliseconds as a floating-point number.*
- `void setclock_delay (t_setclock *x, t_clock *c, long when)`  
*Schedule a Clock on a scheduler.*
- `void setclock_unset (t_setclock *x, t_clock *c)`  
*Remove a Clock from a scheduler.*
- `long setclock_gettime (t_setclock *x)`  
*Find out the current time value of a setclock object.*
- `void setclock_fdelay (t_setclock *x, t_clock *c, double f)`  
*Schedule a Clock on a scheduler, using a floating-point time argument.*
- `void setclock_getftime (t_setclock *x, double *time)`  
*Find out the current time value of a setclock object in floating-point milliseconds.*
- `double systimer_gettime (void)`  
*While most Max timing references "logical" time derived from Max's millisecond scheduler, time values produced by the `systimer_gettime()` are referenced from the CPU clock and can be used to time real world events with microsecond precision.*
- `long gettime (void)`  
*Find out the current logical time of the scheduler in milliseconds.*
- `double gettime_forobject (t_object *x)`  
*Find the correct scheduler for the object and return the current time in milliseconds.*
- `t_scheduler * scheduler_new (void)`  
*Create a new local scheduler.*
- `t_scheduler * scheduler_set (t_scheduler *x)`  
*Make a scheduler current, so that future related calls (such as `clock_delay()`) will affect the appropriate scheduler.*
- `t_scheduler * scheduler_get (void)`  
*Get the currently set scheduler.*

- `t_scheduler * scheduler_fromobject (t_object *obj)`  
*Get the scheduler associated with a given object, if any.*
- `void scheduler_run (t_scheduler *x, double until)`  
*Run scheduler events to a selected time.*
- `void scheduler_settime (t_scheduler *x, double time)`  
*Set the current time of the scheduler.*
- `void scheduler_gettime (t_scheduler *x, double *time)`  
*Retrieve the current time of the selected scheduler.*
- `void scheduler_shift (t_scheduler *x, double amount)`  
*Shift scheduler's current time and run time for all pending clock.*

### 38.46.1 Detailed Description

Clock objects are your interface to Max's scheduler.

To use the scheduler, you create a new Clock object using `clock_new` in your instance creation function. You also have to write a `clock` function that will be executed when the clock goes off, declared as follows:

```
void myobject_tick (myobject *x);
```

The argument `x` is determined by the `arg` argument to `clock_new()`. Almost always it will be pointer to your object. Then, in one of your methods, use `clock_delay()` or `clock_fdelay()` to schedule yourself. If you want unschedule yourself, call `clock_unset()`. To find out what time it is now, use `gettime()` or `clock_gettime()`. More advanced clock operations are possible with the `setclock` object interface described in Chapter 9. We suggest you take advantage of the higher timing precision of the floating-point clock routines—all standard Max 4 timing objects such as metro use them.

When the user has Overdrive mode enabled, your `clock` function will execute at interrupt level.

### 38.46.2 Using Clocks

Under normal circumstances, `gettime` or `clock_gettime` will not be necessary for scheduling purposes if you use `clock_delay` or `clock_fdelay`, but it may be useful for recording the timing of messages or events.

As an example, here's a fragment of how one might go about writing a metronome using the Max scheduler. First, here's the data structure we'll use.

```
typedef struct mymetro {
    t_object *m_obj;
    void *m_clock;
    double m_interval;
    void *m_outlet;
} t_mymetro;
```

We'll assume that the class has been initialized already. Here's the instance creation function that will allocate a new Clock.

```
void *mymetro_create (double defaultInterval)
{
    t_mymetro **x;
    x = (t_mymetro *)newobject (mymetro_class); // allocate space
    x->m_clock = clock_new(x, (method)mymetro_tick); // make a clock
    x->m_interval = defaultInterval; // store the interval
    x->m_outlet = bangout(x); // outlet for ticks
    return x; // return the new object
}
```

Here's the method written to respond to the bang message that starts the metronome.

```
void mymetro_bang (t_mymetro **x)
```

```
{
    clock_fdelay(x->m_clock, 0.);
}
```

Here's the Clock function.

```
void mymetro_tick(t_mymetro *x)
{
    clock_fdelay(x->m_clock, x->m_interval);
    // schedule another metronome tick
    outlet_bang(x->m_outlet); // send out a bang
}
```

You may also want to stop the metronome at some point. Here's a method written to respond to the message stop. It uses `clock_unset`.

```
void mymetro_stop (t_mymetro *x)
{
    clock_unset(x->m_clock);
}
```

In your object's free function, you should call `freeobject` on any Clocks you've created.

```
void mymetro_free (MyMetro **x)
{
    freeobject((t_object *)x->m_clock);
}
```

### 38.46.3 Scheduling with setclock Objects

The setclock object allows a more general way of scheduling Clocks by generalizing the advancement of the time associated with a scheduler. Each setclock object's "time" can be changed by a process other than the internal millisecond clock. In addition, the object implements routines that modify the mapping of the internal millisecond clock onto the current value of time in an object. Your object can call a set of routines that use either setclock or the normal millisecond clock transparently. Many Max objects accept the message `clock` followed by an optional symbol to set their internal scheduling to a named setclock object. The typical implementation passes the binding of a Symbol (the `s_thing` field) to the Setclock functions. By default, the empty symbol is passed. If the binding has been linked to a setclock object, it will be used to schedule the Clock. Otherwise, the Clock is scheduled using the main internal millisecond scheduler. The Setclock data structure is a replacement for `void *` since there will be no reason for external objects to access it directly.

#### 38.46.3.1 Using the setclock Object Routines

Here's an example implementation of the relevant methods of a metronome object using the Setclock routines.

```
typedef struct metro
{
    t_object m_ob;
    long m_interval;
    long m_running;
    void *m_clock;
    t_symbol *m_setclock;
} t_metro;
```

Here's the implementation of the routines for turning the metronome on and off. Assume that in the instance creation function, the `t_symbol` `m_setclock` has been set to the empty symbol (`gensym("")`) and `m_clock` has been created; the `clock` function `metro_tick()` is defined further on.

```
void metro_bang(Metro **x) // turn metronome on
{
    x->m_running = 1;
    setclock_delay(x->m_setclock->s_thing, x->m_clock, 0);
}
void metro_stop(Metro **x)
{
    x->m_running = 0;
    setclock_unset(x->m_setclock->s_thing, x->m_clock);
}
```

Here is the implementation of the clock function metro\_tick() that runs periodically.

```
void metro_tick(Metro **x)
{
    outlet_bang(x->m_obj.o_outlet);
    if (x->m_running)
        setclock_delay(x->m_setclock->s_thing, x->m_clock, x->m_interval);
}
```

Finally, here is an implementation of the method to respond to the clock message. Note that the function tries to verify that a non-zero value bound to the `t_symbol` passed as an argument is in fact an instance of setclock by checking to see if it responds to the unset message. If not, the metronome refuses to assign the `t_symbol` to its internal `m_setclock` field.

```
void metro_clock(Metro *x, t_symbol *s)
{
    void *old = x->m_setclock->s_thing;
    void *c = 0;
    // the line below can be restated as:
    // if s is the empty symbol
    // or s->s_thing is zero
    // or s->s_thing is non-zero and a setclock object
    if ((s == gensym("")) || ((c = s->s_thing) && zgetfn(c, &s_unset)))
    {
        if (old)
            setclock_unset(old, x->m_clock);
        x->m_setclock = s;
        if (x->m_running)
            setclock_delay(c, x->m_clock, 0L);
    }
}
```

### 38.46.4 Creating Schedulers

If you want to schedule events independently of the time of the global Max scheduler, you can create your own scheduler with `scheduler_new()`. By calling `scheduler_set()` with the newly created scheduler, calls to `clock_new()` will create Clocks tied to your scheduler instead of Max's global one. You can then control the time of the scheduler (using `scheduler_settime()`) as well as when it executes clock functions (using `scheduler_run()`). This is a more general facility than the setclock object routines, but unlike using the time from a setclock object to determine when a Clock function runs, once a Clock is tied to a scheduler.

### 38.46.5 Function Documentation

#### 38.46.5.1 `clock_delay()`

```
void clock_delay (
    t_clock * x,
    long n )
```

Schedule the execution of a Clock.

`clock_delay()` sets a clock to go off at a certain number of milliseconds from the current logical time.

##### Parameters

<code>x</code>	Clock to schedule.
<code>n</code>	Delay, in milliseconds, before the Clock will execute.

## See also

[clock\\_fdelay\(\)](#)

**38.46.5.2 clock\_fset()**

```
void clock_fset (
    t_clock * x,
    double when )
```

Schedule the execution of a Clock using a floating-point argument.

[clock\\_delay\(\)](#) sets a clock to go off at a certain number of milliseconds from the current logical time.

## Parameters

<i>c</i>	Clock to schedule.
<i>time</i>	Delay, in milliseconds, before the Clock will execute.

## See also

[clock\\_delay\(\)](#)

**38.46.5.3 clock\_gettime()**

```
void clock_gettime (
    double * time )
```

Find out the current logical time of the scheduler in milliseconds as a floating-point number.

## Parameters

<i>time</i>	Returns the current time.
-------------	---------------------------

## See also

[gettime\(\)](#)  
[setclock\\_gettime\(\)](#)  
[setclock\\_gettime\(\)](#)

### 38.46.5.4 `clock_new()`

```
t_clock* clock_new (
    void * obj,
    method fn )
```

Create a new Clock object.

Normally, `clock_new()` is called in your instance creation function—and it cannot be called from a thread other than the main thread. To get rid of a clock object you created, use `freeobject()`.

#### Parameters

<code>obj</code>	Argument that will be passed to clock function <code>fn</code> when it is called. This will almost always be a pointer to your object.
<code>fn</code>	Function to be called when the clock goes off, declared to take a single argument as shown in <a href="#">Using Clocks</a> .

#### Returns

A pointer to a newly created Clock object.

### 38.46.5.5 `clock_unset()`

```
void clock_unset (
    t_clock * x )
```

Cancel the scheduled execution of a Clock.

`clock_unset()` will do nothing (and not complain) if the Clock passed to it has not been set.

#### Parameters

<code>x</code>	Clock to cancel.
----------------	------------------

### 38.46.5.6 `gettime()`

```
long gettime (
    void )
```

Find out the current logical time of the scheduler in milliseconds.

**Returns**

Returns the current time.

**See also**

[clock\\_gettime\(\)](#)

**38.46.5.7 gettime\_forobject()**

```
double gettime_forobject (
    t_object * x )
```

Find the correct scheduler for the object and return the current time in milliseconds.

**Returns**

Returns the current time.

**See also**

[clock\\_gettime\(\)](#)

**38.46.5.8 scheduler\_fromobject()**

```
t_scheduler* scheduler_fromobject (
    t_object * obj )
```

Get the scheduler associated with a given object, if any.

**Parameters**

<i>o</i>	The object who's scheduler is to be returned.
----------	---

**Returns**

This routine returns a pointer to the scheduler or the passed in object,

**See also**

[Creating Schedulers](#)

### 38.46.5.9 scheduler\_get()

```
t_scheduler* scheduler_get (
    void )
```

Get the currently set scheduler.

#### Returns

This routine returns a pointer to the current scheduler,

#### See also

[Creating Schedulers](#)

### 38.46.5.10 scheduler\_gettime()

```
void scheduler_gettime (
    t_scheduler * x,
    double * time )
```

Retrieve the current time of the selected scheduler.

#### Parameters

<i>x</i>	The scheduler to query.
<i>time</i>	The current time of the selected scheduler.

#### See also

[Creating Schedulers](#)

### 38.46.5.11 scheduler\_new()

```
t_scheduler* scheduler_new (
    void )
```

Create a new local scheduler.

#### Returns

A pointer to the newly created scheduler.

**See also**

[Creating Schedulers](#)

**38.46.5.12 scheduler\_run()**

```
void scheduler_run (
    t_scheduler * x,
    double until )
```

Run scheduler events to a selected time.

**Parameters**

<i>x</i>	The scheduler to advance.
<i>until</i>	The ending time for this run (in milliseconds).

**See also**

[Creating Schedulers](#)

**38.46.5.13 scheduler\_set()**

```
t_scheduler* scheduler_set (
    t_scheduler * x )
```

Make a scheduler current, so that future related calls (such as [clock\\_delay\(\)](#)) will affect the appropriate scheduler.

**Parameters**

<i>x</i>	The scheduler to make current.
----------	--------------------------------

**Returns**

This routine returns a pointer to the previously current scheduler, saved and restored when local scheduling is complete.

**See also**

[Creating Schedulers](#)

### 38.46.5.14 scheduler\_settime()

```
void scheduler_settime (
    t_scheduler * x,
    double time )
```

Set the current time of the scheduler.

#### Parameters

<i>x</i>	The scheduler to set.
<i>time</i>	The new current time for the selected scheduler (in milliseconds).

#### See also

[Creating Schedulers](#)

### 38.46.5.15 scheduler\_shift()

```
void scheduler_shift (
    t_scheduler * x,
    double amount )
```

Shift scheduler's current time and run time for all pending clock.

Could be used to change scheduler's time reference without impacting current clocks.

#### Parameters

<i>x</i>	The scheduler to affect.
<i>amount</i>	Number of milliseconds to shift by.

#### See also

[Creating Schedulers](#)

### 38.46.5.16 setclock\_delay()

```
void setclock_delay (
    t_setclock * x,
```

```
t_clock * c,
long when )
```

Schedule a Clock on a scheduler.

Schedules the Clock c to execute at time units after the current time. If scheduler x is 0 or does not point to a setclock object, the internal millisecond scheduler is used. Otherwise c is scheduled on the setclock object's list of Clocks. The Clock should be created with [clock\\_new\(\)](#), the same as for a Clock passed to [clock\\_delay\(\)](#).

#### Parameters

<i>x</i>	A setclock object to be used for scheduling this clock.
<i>c</i>	Clock object containing the function to be executed.
<i>time</i>	Time delay (in the units of the Setclock) from the current time when the Clock will be executed.

#### See also

[Scheduling with setclock Objects](#)

[setclock\\_fdelay\(\)](#)

### 38.46.5.17 setclock\_fdelay()

```
void setclock_fdelay (
    t_setclock * x,
    t_clock * c,
    double f )
```

Schedule a Clock on a scheduler, using a floating-point time argument.

#### Parameters

<i>s</i>	A setclock object to be used for scheduling this clock.
<i>c</i>	Clock object containing the function to be executed.
<i>time</i>	Time delay (in the units of the Setclock) from the current time when the Clock will be executed.

#### See also

[Scheduling with setclock Objects](#)

[setclock\\_delay\(\)](#)

### 38.46.5.18 `setclock_gettime()`

```
void setclock_gettime (
    t_setclock * x,
    double * time )
```

Find out the current time value of a setclock object in floating-point milliseconds.

#### Parameters

<i>s</i>	A setclock object.
<i>time</i>	The current time in milliseconds.

#### See also

[Scheduling with setclock Objects](#)

[setclock\\_gettime\(\)](#)

### 38.46.5.19 `setclock_gettime()`

```
long setclock_gettime (
    t_setclock * x )
```

Find out the current time value of a setclock object.

#### Parameters

<i>x</i>	A setclock object.
----------	--------------------

#### Returns

Returns the current time value of the setclock object scheduler. If scheduler is 0, setclock\_gettime is equivalent to the function gettimeofday that returns the current value of the internal millisecond clock.

#### See also

[Scheduling with setclock Objects](#)

[setclock\\_gettime\(\)](#)

### 38.46.5.20 setclock\_unset()

```
void setclock_unset (
    t_setclock * x,
    t_clock * c )
```

Remove a Clock from a scheduler.

This function unschedules the Clock c in the list of Clocks in the setclock object x, or the internal millisecond scheduler if scheduler is 0.

#### Parameters

x	The setclock object that was used to schedule this clock. If 0, the clock is unscheduled from the internal millisecond scheduler.
c	Clock object to be removed from the scheduler.

#### See also

[Scheduling with setclock Objects](#)

### 38.46.5.21 systimer\_gettime()

```
double systimer_gettime (
    void )
```

While most Max timing references "logical" time derived from Max's millisecond scheduler, time values produced by the [systimer\\_gettime\(\)](#) are referenced from the CPU clock and can be used to time real world events with microsecond precision.

The standard 'cpuclock' external in Max is a simple wrapper around this function.

#### Returns

Returns the current real-world time.

## 38.47 Qelems

Your object's methods may be called at interrupt level.

Collaboration diagram for Qelems:



## Functions

- void \* [qelem\\_new](#) (void \*obj, method fn)  
*Create a new Qelem.*
- void [qelem\\_set](#) (t\_qelem \*x)  
*Cause a Qelem to execute.*
- void [qelem\\_unset](#) (t\_qelem \*x)  
*Cancel a Qelem's execution.*
- void [qelem\\_free](#) (t\_qelem \*x)  
*Free a Qelem object created with [qelem\\_new\(\)](#).*
- void [qelem\\_front](#) (t\_qelem \*x)  
*Cause a Qelem to execute with a higher priority.*

### 38.47.1 Detailed Description

Your object's methods may be called at interrupt level.

This happens when the user has Overdrive mode enabled and one of your methods is called, directly or indirectly, from a scheduler Clock function. This means that you cannot count on doing certain things—like drawing, asking the user what file they would like opened, or calling any Macintosh toolbox trap that allocates or purges memory—from within any method that responds to any message that could be sent directly from another Max object. The mechanism you'll use to get around this limitation is the Qelem (queue element) structure. Qelems also allow processor-intensive tasks to be done at a lower priority than in an interrupt. As an example, drawing on the screen, especially in color, takes a long time in comparison with a task like sending MIDI data.

A Qelem works very much like a Clock. You create a new Qelem in your creation function with [qelem\\_new](#) and store a pointer to it in your object. Then you write a queue function, very much like the clock function (it takes the same single argument that will usually be a pointer to your object) that will be called when the Qelem has been set. You set the Qelem to run its function by calling [qelem\\_set\(\)](#).

Often you'll want to use Qelems and Clocks together. For example, suppose you want to update the display for a counter that changes 20 times a second. This can be accomplished by writing a Clock function that calls [qelem\\_set\(\)](#) and then reschedules itself for 50 milliseconds later using the technique shown in the metronome example above. This scheme works even if you call [qelem\\_set\(\)](#) faster than the computer can draw the counter, because if a Qelem is already set, [qelem\\_set\(\)](#) will not set it again. However, when drawing the counter, you'll display its current value, not a specific value generated in the Clock function.

Note that the Qelem-based defer mechanism discussed later in this chapter may be easier for lowering the priority of one-time events, such as opening a standard file dialog box in response to a read message.

If your Qelem routine sends messages using [outlet\\_int\(\)](#) or any other of the outlet functions, it needs to use the lockout mechanism described in the Interrupt Level Considerations section.

### 38.47.2 Function Documentation

#### 38.47.2.1 [qelem\\_free\(\)](#)

```
void qelem_free (
    t_qelem * x )
```

Free a Qelem object created with [qelem\\_new\(\)](#).

Typically this will be in your object's free function.

**Parameters**

x	The Qelem to destroy.
---	-----------------------

**38.47.2.2 qelem\_front()**

```
void qelem_front (
    t_qelem * x )
```

Cause a Qelem to execute with a higher priority.

This function is identical to [qelem\\_set\(\)](#), except that the Qelem's function is placed at the front of the list of routines to execute in the main thread instead of the back. Be polite and only use [qelem\\_front\(\)](#) only for special time-critical applications.

**Parameters**

x	The Qelem whose function will be executed in the main thread.
---	---

**38.47.2.3 qelem\_new()**

```
void* qelem_new (
    void * obj,
    method fn )
```

Create a new Qelem.

The created Qelem will need to be freed using [qelem\\_free\(\)](#), do not use [freeobject\(\)](#).

**Parameters**

obj	Argument to be passed to function fun when the Qelem executes. Normally a pointer to your object.
fn	Function to execute.

**Returns**

A pointer to a Qelem instance. You need to store this value to pass to [qelem\\_set\(\)](#).

**Remarks**

Any kind of drawing or calling of Macintosh Toolbox routines that allocate or purge memory should be done in a Qelem function.

#### 38.47.2.4 qelem\_set()

```
void qelem_set (
    t_qelem * x )
```

Cause a Qelem to execute.

##### Parameters

<i>q</i>	The Qelem whose function will be executed in the main thread.
----------	---

##### Remarks

The key behavior of [qelem\\_set\(\)](#) is this: if the Qelem object has already been set, it will not be set again. (If this is not what you want, see [defer\(\)](#).) This is useful if you want to redraw the state of some data when it changes, but not in response to changes that occur faster than can be drawn. A Qelem object is unset after its queue function has been called.

#### 38.47.2.5 qelem\_unset()

```
void qelem_unset (
    t_qelem * x )
```

Cancel a Qelem's execution.

If the Qelem's function is set to be called, [qelem\\_unset\(\)](#) will stop it from being called. Otherwise, [qelem\\_unset\(\)](#) does nothing.

##### Parameters

<i>q</i>	The Qelem whose execution you wish to cancel.
----------	---

## 38.48 Systime API

The Systime API provides the means of getting the system time, instead of the scheduler time as you would with functions like [gettime\(\)](#).

Collaboration diagram for Systime API:



## Data Structures

- struct [t\\_datetime](#)

*The Systime data structure.*

## Enumerations

- enum [e\\_max\\_dateflags](#) { [SYSDATEFORMAT\\_FLAGS\\_SHORT](#) , [SYSDATEFORMAT\\_FLAGS\\_MEDIUM](#) , [SYSDATEFORMAT\\_FLAGS\\_LONG](#) , [SYSDATEFORMAT\\_RELATIVE](#) }

*Flags for the [sysdateformat\\_formatdatetime\(\)](#) function.*

## Functions

- [t\\_uint32 systime\\_ticks](#) (void)

*Find out the operating system's time in ticks.*

- [t\\_uint32 systime\\_ms](#) (void)

*Find out the operating system's time in milliseconds.*

- [t\\_int64 systime\\_datetime\\_milliseconds](#) (void)

*Find out the current date/time as number of ms since January 1, 1970, GMT.*

- void [systime\\_datetime](#) ([t\\_datetime](#) \*d)

*Find out the operating system's current local date and time.*

- [t\\_ptr\\_uint systime\\_seconds](#) (void)

*Find out the operating system's time in seconds since midnight, January 1, 1904, GMT (mac HFS time).*

- void [systime\\_secondstodate](#) ([t\\_ptr\\_uint](#) secs, [t\\_datetime](#) \*d)

*Convert a time in seconds into a [t\\_datetime](#) representation.*

- [t\\_ptr\\_uint systime\\_datetoseconds](#) (const [t\\_datetime](#) \*d)

*Convert a [t\\_datetime](#) representation of time into seconds since midnight, January 1, 1904, GMT.*

- void [sysdateformat\\_strftimetodatetime](#) (const char \*strf, [t\\_datetime](#) \*d)

*Fill a [t\\_datetime](#) struct with a datetime formatted string.*

- void [sysdateformat\\_formatdatetime](#) (const [t\\_datetime](#) \*d, long dateflags, long timeflags, char \*s, long buflen)

*Get a human friendly string representation of a [t\\_datetime](#).*

### 38.48.1 Detailed Description

The Systime API provides the means of getting the system time, instead of the scheduler time as you would with functions like [gettime\(\)](#).

### 38.48.2 Enumeration Type Documentation

#### 38.48.2.1 e\_max\_dateflags

```
enum e_max_dateflags
```

Flags for the [sysdateformat\\_formatdatetime\(\)](#) function.

Enumerator

SYSDATEFORMAT_FLAGS_SHORT	short
SYSDATEFORMAT_FLAGS_MEDIUM	medium
SYSDATEFORMAT_FLAGS_LONG	long

### 38.48.3 Function Documentation

#### 38.48.3.1 sysdateformat\_formatdatetime()

```
void sysdateformat_formatdatetime (
    const t_datetime * d,
    long dateflags,
    long timeflags,
    char * s,
    long buflen )
```

Get a human friendly string representation of a [t\\_datetime](#).

For example: "Dec 17, 2020 at 10:48 AM" or, when dateflags is SYSDATEFORMAT\_RELATIVE a string like "Today", "Yesterday", etc.

Parameters

<i>d</i>	The address of a <a href="#">t_datetime</a> to format as a string. The <a href="#">t_datetime</a> entries correspond to the local time.
<i>dateflags</i>	One of the values defined in <a href="#">e_max_dateflags</a> .
<i>timeflags</i>	Currently unused. Pass 0.
<i>s</i>	An already allocated string to hold the human friendly result.
<small>Cycling 74 <i>buflen</i></small>	The number of characters allocated to the string <i>s</i> .

### 38.48.3.2 sysdateformat\_strftimetodatetime()

```
void sysdateformat_strftimetodatetime (
    const char * strf,
    t_datetime * d )
```

Fill a [t\\_datetime](#) struct with a datetime formatted string.

For example, the string "2007-12-24 12:21:00".

#### Parameters

<i>strf</i>	A string containing the datetime.
<i>d</i>	The address of a <a href="#">t_datetime</a> to fill.

### 38.48.3.3 systime\_datetime()

```
void systime_datetime (
    t_datetime * d )
```

Find out the operating system's current local date and time.

#### Parameters

<i>d</i>	Returns the system's date and time in the local time zone in a <a href="#">t_datetime</a> data structure.
----------	---

### 38.48.3.4 systime\_datetime\_milliseconds()

```
t_int64 systime_datetime_milliseconds (
    void )
```

Find out the current date/time as number of ms since January 1, 1970, GMT.

#### Returns

the number of milliseconds since January 1, 1970, GMT.

### 38.48.3.5 systime\_datetoseconds()

```
t_ptr_uint systime_datetoseconds (
    const t_datetime * d )
```

Convert a [t\\_datetime](#) representation of time into seconds since midnight, January 1, 1904, GMT.

#### Parameters

<i>d</i>	The address of a <a href="#">t_datetime</a> to be converted to seconds. The <a href="#">t_datetime</a> values are in the local time zone.
----------	---

#### Returns

The number of seconds between midnight, January 1, 1904, GMT and the time represented in *d*.

### 38.48.3.6 systime\_ms()

```
t_uint32 systime_ms (
    void )
```

Find out the operating system's time in milliseconds.

Note that this is approximately the number of milliseconds since the OS was started up.

#### Returns

the system time in milliseconds.

### 38.48.3.7 systime\_seconds()

```
t_ptr_uint systime_seconds (
    void )
```

Find out the operating system's time in seconds since midnight, January 1, 1904, GMT (mac HFS time).

#### Returns

the system time in seconds since midnight, January 1, 1904, GMT.

### 38.48.3.8 systime\_secondstodate()

```
void systime_secondstodate (
    t_ptr_uint secs,
    t_datetime * d )
```

Convert a time in seconds into a [t\\_datetime](#) representation.

**Parameters**

<code>secs</code>	A number of seconds since midnight, January 1, 1904, GMT, to be represented as a <a href="#">t_datetime</a> .
<code>d</code>	The address of a <a href="#">t_datetime</a> that will be filled to the corresponding time, in the local time zone.

**38.48.3.9 systime\_ticks()**

```
t_uint32 systime_ticks (
    void )
```

Find out the operating system's time in ticks.

**Returns**

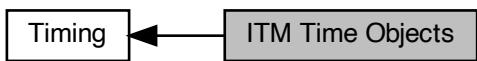
the system time in ticks.

Referenced by `jit_rand_setseed()`.

**38.49 ITM Time Objects**

ITM Time Objects are a high-level interface to ITM, a tempo-based scheduler API.

Collaboration diagram for ITM Time Objects:

**Typedefs**

- `typedef typedefBEGIN_USING_C_LINKAGE struct _itm t_itm`  
*A low-level object for tempo-based scheduling.*

## Enumerations

```
• enum {
    TIME_FLAGS_LOCATION, TIME_FLAGS_TICKSONLY, TIME_FLAGS_FIXEDONLY, TIME_FLAGS_LOOKAHEAD
    ,
    TIME_FLAGS_USECLOCK, TIME_FLAGS_USEQELEM, TIME_FLAGS_FIXED, TIME_FLAGS_PERMANENT
    ,
    TIME_FLAGS_TRANSPORT, TIME_FLAGS_EVENTLIST, TIME_FLAGS_CHECKSCHEDULE, TIME_FLAGS_LISTENTICKS
    ,
    TIME_FLAGS_NOUNITS, TIME_FLAGS_BBUSOURCE, TIME_FLAGS_POSITIVE }
```

*Flags that determine attribute and time object behavior.*

## Functions

- void \* **itm\_getglobal** (void)
 

*Return the global (default / unnamed) itm object.*
- void \* **itm\_getnamed** (**t\_symbol** \*s, void \*scheduler, **t\_symbol** \*defaultclocksourcename, long create)
 

*Return a named itm object.*
- void **itm\_reference** (**t\_itm** \*x)
 

*Reference an itm object.*
- void **itm\_dereference** (**t\_itm** \*x)
 

*Stop referencing an itm object.*
- double **itm\_gettime** (**t\_itm** \*x)
 

*Report the current internal time.*
- double **itm\_getticks** (**t\_itm** \*x)
 

*Report the current time of the itm in ticks.*
- void **itm\_dump** (**t\_itm** \*x)
 

*Print diagnostic information about an itm object to the Max window.*
- void **itm\_settimesignature** (**t\_itm** \*x, long num, long denom, long flags)
 

*Set an itm object's current time signature.*
- void **itm\_gettimesignature** (**t\_itm** \*x, long \*num, long \*denom)
 

*Query an itm object for its current time signature.*
- void **itm\_pause** (**t\_itm** \*x)
 

*Pause the passage of time for an itm object.*
- void **itm\_resume** (**t\_itm** \*x)
 

*Start the passage of time for an itm object, from it's current location.*
- long **itm\_getstate** (**t\_itm** \*x)
 

*Find out if time is currently progressing for a given itm object.*
- void **itm\_setresolution** (double res)
 

*Set the number of ticks-per-quarter-note globally for the itm system.*
- double **itm\_getresolution** (void)
 

*Get the number of ticks-per-quarter-note globally from the itm system.*
- **t\_symbol** \* **itm\_getname** (**t\_itm** \*x)
 

*Given an itm object, return its name.*
- double **itm\_tickstoms** (**t\_itm** \*\*x, double ticks)
 

*Convert a time value in ticks to the equivalent value in milliseconds, given the context of a specified itm object.*
- double **itm\_mstoticks** (**t\_itm** \*\*x, double ms)

- `double itm_mstosamps (t_itm *x, double ms)`

*Convert a time value in milliseconds to the equivalent value in ticks, given the context of a specified itm object.*
- `double itm_sampstoms (t_itm *x, double samps)`

*Convert a time value in milliseconds to the equivalent value in samples, given the context of a specified itm object.*
- `void itm_barbeatunitstoticks (t_itm *x, long bars, long beats, double *ticks, char position)`

*Convert a time value in bbu to the equivalent value in ticks, given the context of a specified itm object.*
- `void itm_ticksbarbeatunits (t_itm *x, double ticks, long *bars, long *beats, double *units, char position)`

*Convert a time value in bbu to the equivalent value in ticks, given the context of a specified itm object.*
- `long itm_isunitfixed (t_symbol *u)`

*Given the name of a time unit (e.g.*
- `void time_stop (t_timeobject *x)`

*Stop a currently scheduled time object.*
- `void time_tick (t_timeobject *x)`

*Execute a time object's task, then if it was already set to execute, reschedule for the current interval value of the object.*
- `double time_getms (t_timeobject *x)`

*Convert the value of a time object to milliseconds.*
- `double time_getticks (t_timeobject *x)`

*Convert the value of a time object to ticks.*
- `void time_getphase (t_timeobject *tx, double *phase, double *slope, double *ticks)`

*Return the phase of the ITM object (transport) associated with a time object.*
- `void time_listen (t_timeobject *x, t_symbol *attr, long flags)`

*Specify that a millisecond-based attribute to be updated automatically when the converted milliseconds of the time object's value changes.*
- `void time_setvalue (t_timeobject *tx, t_symbol *s, long argc, t_atom *argv)`

*Set the current value of a time object (either an interval or a position) using a Max message.*
- `void class_time_addattr (t_class *c, const char *attrname, const char *attrlabel, long flags)`

*Create an attribute permitting a time object to be changed in a user-friendly way.*
- `void *time_new (t_object *owner, t_symbol *attrname, method tick, long flags)`

*Create a new time object.*
- `t_object *time_getnamed (t_object *owner, t_symbol *attrname)`

*Return a time object associated with an attribute of an owning object.*
- `long time_isfixedunit (t_timeobject *x)`

*Return whether this time object currently holds a fixed (millisecond-based) value.*
- `void time_schedule (t_timeobject *x, t_timeobject *quantize)`

*Schedule a task, with optional quantization.*
- `void time_schedule_limit (t_timeobject *x, t_timeobject *quantize)`

*Schedule a task, with optional minimum interval.,*
- `void time_now (t_timeobject *x, t_timeobject *quantize)`

*Schedule a task for right now, with optional quantization.*
- `void *time_getitm (t_timeobject *ox)`

*Return the ITM object associated with this time object.*
- `double time_calcquantize (t_timeobject *ox, t_itm *vitm, t_timeobject *oq)`

*Calculate the quantized interval (in ticks) if this time object were to be scheduled at the current time.*
- `void time_setclock (t_timeobject *tx, t_symbol *sc)`

*Associate a named setclock object with a time object (unsupported).*

## Variables

- [BEGIN\\_USING\\_C\\_LINKAGE](#) `typedef t_object t_timeobject`  
*A high-level time object for tempo-based scheduling.*

### 38.49.1 Detailed Description

ITM Time Objects are a high-level interface to ITM, a tempo-based scheduler API.

They provide an abstraction so your object can schedule events either in milliseconds (as traditional clock objects) or ticks (tempo-relative units).

### 38.49.2 Typedef Documentation

#### 38.49.2.1 `t_itm`

```
typedef typedefBEGIN_USING_C_LINKAGE struct _itm t_itm
```

A low-level object for tempo-based scheduling.

See also

[t\\_timeobject](#)

[ITM](#)

### 38.49.3 Enumeration Type Documentation

**Enumerator****38.49.3.1 anonymous enum**

```
anonymous enum
```

Flags that determine attribute and time object behavior.

**Enumerator**

TIME_FLAGS_LOCATION	1 1 0 location-based bar/beat/unit values (as opposed to interval values, which are 0 0 0 relative)
TIME_FLAGS_TICKSONLY	only ticks-based values (not ms) are acceptable
TIME_FLAGS_FIXEDONLY	only fixed values (ms, hz, samples) are acceptable
TIME_FLAGS_LOOKAHEAD	add lookahead attribute (unsupported)
TIME_FLAGS_USECLOCK	this time object will schedule events, not just hold a value
TIME_FLAGS_USEQELEM	this time object will defer execution of scheduled events to low priority thread
TIME_FLAGS_FIXED	will only use normal clock (i.e., will never execute out of ITM)
TIME_FLAGS_PERMANENT	event will be scheduled in the permanent list (tied to a specific time)
TIME_FLAGS_TRANSPORT	add a transport attribute
TIME_FLAGS_EVENTLIST	add an eventlist attribute (unsupported)
TIME_FLAGS_CHECKSCHEDULE	internal use only
TIME_FLAGS_LISTENTICKS	flag for time_listen: only get notifications if the time object holds tempo-relative values
TIME_FLAGS_NOUNITS	internal use only
TIME_FLAGS_BBUSOURCE	source time was in bar/beat/unit values, need to recalculate when time sig changes
TIME_FLAGS_POSITIVE	constrain any values <= 0 to a minimum value (default: 0)

**38.49.4 Function Documentation****38.49.4.1 class\_time\_addattr()**

```
void class_time_addattr (
    t_class * c,
    const char * attrname,
    const char * attrlabel,
    long flags )
```

Create an attribute permitting a time object to be changed in a user-friendly way.

**Parameters**

<i>c</i>	Class being initialized.
<i>attrname</i>	Name of the attribute associated with the time object.
<i>attrlabel</i>	Descriptive label for the attribute (appears in the inspector)
<i>flags</i>	Options, see "Flags that determine time object behavior" above

**38.49.4.2 item\_barbeatunitstoticks()**

```
void item_barbeatunitstoticks (
    t_item * x,
    long bars,
    long beats,
    double units,
    double * ticks,
    char position )
```

Convert a time value in bbu to the equivalent value in ticks, given the context of a specified item object.

**Parameters**

<i>x</i>	An item object.
<i>bars</i>	The measure number of the location/position.
<i>beats</i>	The beat number of the location/position.
<i>units</i>	The number of ticks past the beat of the location/position.
<i>ticks</i>	The address of a variable to hold the number of ticks upon return.
<i>position</i>	Set this parameter to <a href="#">TIME_FLAGS_LOCATION</a> or to zero (for position mode).

**38.49.4.3 item\_dereference()**

```
void item_dereference (
    t_item * x )
```

Stop referencing an item object.

When you are done using an item object, you must call this function to decrement its reference count.

**Parameters**

<i>x</i>	The item object.
----------	------------------

#### 38.49.4.4 `itm_dump()`

```
void itm_dump (
    t_itm * x )
```

Print diagnostic information about an item object to the Max window.

##### Parameters

x	The item object.
---	------------------

#### 38.49.4.5 `itm_getglobal()`

```
void* itm_getglobal (
    void )
```

Return the global (default / unnamed) item object.

##### Returns

The global `t_itm` object.

#### 38.49.4.6 `itm_getname()`

```
t_symbol* itm_getname (
    t_itm * x )
```

Given an item object, return its name.

##### Parameters

x	The item object.
---	------------------

##### Returns

The name of the item.

**38.49.4.7 item\_getnamed()**

```
void* item_getnamed (
    t_symbol * s,
    void * scheduler,
    t_symbol * defaultclocksourcename,
    long create )
```

Return a named item object.

**Parameters**

<i>s</i>	The name of the item to return.
<i>scheduler</i>	
<i>defaultclocksourcename</i>	
<i>create</i>	If non-zero, then create this named item should it not already exist.

**Returns**

The global [t\\_item](#) object.

**38.49.4.8 item\_getresolution()**

```
double item_getresolution (
    void )
```

Get the number of ticks-per-quarter-note globally from the item system.

**Returns**

The number of ticks-per-quarter-note.

**See also**

[item\\_setresolution\(\)](#)

**38.49.4.9 item\_getstate()**

```
long item_getstate (
    t_item * x )
```

Find out if time is currently progressing for a given item object.

**Parameters**

x	The item object.
---	------------------

**Returns**

Returns non-zero if the time is running, or zero if it is paused.

**See also**

[itm\\_pause\(\)](#)

[itm\\_resume\(\)](#)

**38.49.4.10 itm\_getticks()**

```
double itm_getticks (
    t_itm * x )
```

Report the current time of the item in ticks.

You can use functions such as [itm\\_ticksstobarbeatunits\(\)](#) or [itm\\_ticksstoms\(\)](#) to convert to a different representation of the time.

**Parameters**

x	The item object.
---	------------------

**Returns**

The current time in ticks.

**38.49.4.11 itm\_gettime()**

```
double itm_gettime (
    t_itm * x )
```

Report the current internal time.

This is the same as calling [clock\\_gettime\(\)](#);

**Parameters**

<i>x</i>	The item object.
----------	------------------

**Returns**

The current internal time.

**38.49.4.12 item\_gettimesignature()**

```
void item_gettimesignature (
    t_item * x,
    long * num,
    long * denom )
```

Query an item object for its current time signature.

**Parameters**

<i>x</i>	The item object.
<i>num</i>	The address of a variable to hold the top number of the time signature upon return.
<i>denom</i>	The address of a variable to hold the bottom number of the time signature upon return.

**38.49.4.13 item\_isunitfixed()**

```
long item_isunitfixed (
    t_symbol * u )
```

Given the name of a time unit (e.g.

'ms', 'ticks', 'bbu', 'samples', etc.), determine whether the unit is fixed (doesn't change with tempo, time-signature, etc.) or whether it is flexible.

**Parameters**

<i>u</i>	The name of the time unit.
----------	----------------------------

**Returns**

Zero if the unit is fixed (milliseconds, for example) or non-zero if it is flexible (ticks, for example).

#### 38.49.4.14 `itm_mstosamps()`

```
double itm_mstosamps (
    t_itm * x,
    double ms )
```

Convert a time value in milliseconds to the equivalent value in samples, given the context of a specified itm object.

##### Parameters

<code>x</code>	An itm object.
<code>ms</code>	A time specified in ms.

##### Returns

The time specified in samples.

#### 38.49.4.15 `itm_mstoticks()`

```
double itm_mstoticks (
    t_itm * x,
    double ms )
```

Convert a time value in milliseconds to the equivalent value in ticks, given the context of a specified itm object.

##### Parameters

<code>x</code>	An itm object.
<code>ms</code>	A time specified in ms.

##### Returns

The time specified in ticks.

#### 38.49.4.16 `itm_pause()`

```
void itm_pause (
    t_itm * x )
```

Pause the passage of time for an itm object.

This is the equivalent to setting the state of a transport object to 0 with a toggle.

**Parameters**

x	The item object.
---	------------------

**38.49.4.17 itm\_reference()**

```
void itm_reference (
    t_itm * x )
```

Reference an item object.

When you are using an item object, you must call this function to increment its reference count.

**Parameters**

x	The item object.
---	------------------

**38.49.4.18 itm\_resume()**

```
void itm_resume (
    t_itm * x )
```

Start the passage of time for an item object, from it's current location.

This is the equivalent to setting the state of a transport object to 0 with a toggle.

**Parameters**

x	The item object.
---	------------------

**38.49.4.19 itm\_sampstoms()**

```
double itm_sampstoms (
    t_itm * x,
    double samps )
```

Convert a time value in samples to the equivalent value in milliseconds, given the context of a specified item object.

**Parameters**

<i>x</i>	An item object.
<i>samps</i>	A time specified in samples.

**Returns**

The time specified in ms.

**38.49.4.20 `itm_setresolution()`**

```
void itm_setresolution (
    double res )
```

Set the number of ticks-per-quarter-note globally for the item system.

The default is 480.

**Parameters**

<i>res</i>	The number of ticks-per-quarter-note.
------------	---------------------------------------

**See also**

[itm\\_getresolution\(\)](#)

**38.49.4.21 `itm_settimesignature()`**

```
void itm_settimesignature (
    t_itm * x,
    long num,
    long denom,
    long flags )
```

Set an item object's current time signature.

**Parameters**

<i>x</i>	The item object.
<i>num</i>	The top number of the time signature.
<i>denom</i>	The bottom number of the time signature.
<i>flags</i>	Currently unused – pass zero.

#### 38.49.4.22 `itm_ticksbarbeatunits()`

```
void itm_ticksbarbeatunits (
    t_itm * x,
    double ticks,
    long * bars,
    long * beats,
    double * units,
    char position )
```

Convert a time value in bbu to the equivalent value in ticks, given the context of a specified item object.

##### Parameters

<i>x</i>	An item object.
<i>ticks</i>	The number of ticks to translate into a time represented as bars, beats, and ticks.
<i>bars</i>	The address of a variable to hold the measure number of the location/position upon return.
<i>beats</i>	The address of a variable to hold the beat number of the location/position upon return.
<i>units</i>	The address of a variable to hold the number of ticks past the beat of the location/position upon return.
<i>position</i>	Set this parameter to <a href="#">TIME_FLAGS_LOCATION</a> or to zero (for position mode).

#### 38.49.4.23 `itm_ticksoms()`

```
double itm_ticksoms (
    t_itm * x,
    double ticks )
```

Convert a time value in ticks to the equivalent value in milliseconds, given the context of a specified item object.

##### Parameters

<i>x</i>	An item object.
<i>ticks</i>	A time specified in ticks.

##### Returns

The time specified in ms.

### 38.49.4.24 time\_calcquantize()

```
double time_calcquantize (
    t_timeobject * ox,
    t_itm * vitm,
    t_timeobject * oq )
```

Calculate the quantized interval (in ticks) if this time object were to be scheduled at the current time.

#### Parameters

<i>ox</i>	Time object.
<i>vitm</i>	The associated ITM object (use <a href="#">time_getitm()</a> to determine it).
<i>oq</i>	A time object that holds a quantization interval, can be NULL.

#### Returns

Interval (in ticks) for scheduling this object.

### 38.49.4.25 time\_getitm()

```
void* time_getitm (
    t_timeobject * ox )
```

Return the ITM object associated with this time object.

#### Parameters

<i>ox</i>	Time object.
-----------	--------------

#### Returns

The associated [t\\_itm](#) object.

### 38.49.4.26 time\_getms()

```
double time_getms (
    t_timeobject * x )
```

Convert the value of a time object to milliseconds.

**Parameters**

x	The time object.
---	------------------

**Returns**

The time object's value, converted to milliseconds.

**38.49.4.27 time\_getnamed()**

```
t_object* time_getnamed (
    t_object * owner,
    t_symbol * attrname )
```

Return a time object associated with an attribute of an owning object.

**Parameters**

owner	Object that owns this time object (task routine, if any, will pass owner as argument).
attrname	Name of the attribute associated with the time object.

**Returns**

The [t\\_timeobject](#) associated with the named attribute.

**38.49.4.28 time\_getphase()**

```
void time_getphase (
    t_timeobject * tx,
    double * phase,
    double * slope,
    double * ticks )
```

Return the phase of the ITM object (transport) associated with a time object.

**Parameters**

tx	The time object.
phase	Pointer to a double to receive the progress within the specified time value of the associated ITM object.
slope	Pointer to a double to receive the slope (phase difference) within the specified time value of the associated ITM object.
ticks	

### 38.49.4.29 time\_getticks()

```
double time_getticks (
    t_timeobject * x )
```

Convert the value of a time object to ticks.

#### Parameters

x	The time object.
---	------------------

#### Returns

The time object's value, converted to ticks.

### 38.49.4.30 time\_isfixedunit()

```
long time_isfixedunit (
    t_timeobject * x )
```

Return whether this time object currently holds a fixed (millisecond-based) value.

#### Parameters

x	Time object.
---	--------------

#### Returns

True if time object's current value is fixed, false if it is tempo-relative.

### 38.49.4.31 time\_listen()

```
void time_listen (
    t_timeobject * x,
    t_symbol * attr,
    long flags )
```

Specify that a millisecond-based attribute to be updated automatically when the converted milliseconds of the time object's value changes.

**Parameters**

<i>x</i>	The time object.
<i>attr</i>	Name of the millisecond based attribute in the owning object that will be updated
<i>flags</i>	If TIME_FLAGS_LISTENTICKS is passed here, updating will not happen if the time value is fixed (ms) based

**38.49.4.32 time\_new()**

```
void* time_new (
    t_object * owner,
    t_symbol * attrname,
    method tick,
    long flags )
```

Create a new time object.

**Parameters**

<i>owner</i>	Object that will own this time object (task routine, if any, will pass owner as argument).
<i>attrname</i>	Name of the attribute associated with the time object.
<i>tick</i>	Task routine that will be executed (can be NULL)
<i>flags</i>	Options, see "Flags that determine time object behavior" above

**Returns**

The newly created [t\\_timeobject](#).

**38.49.4.33 time\_now()**

```
void time_now (
    t_timeobject * x,
    t_timeobject * quantize )
```

Schedule a task for right now, with optional quantization.

**Parameters**

<i>x</i>	The time object that schedules temporary events. The time interval is ignored and 0 ticks is used instead.
<i>quantize</i>	A time object that holds a quantization interval, can be NULL for no quantization.

### 38.49.4.34 time\_schedule()

```
void time_schedule (
    t_timeobject * x,
    t_timeobject * quantize )
```

Schedule a task, with optional quantization.

#### Parameters

<i>x</i>	The time object that schedules temporary events (must have been created with TIME_FLAGS_USECLOCK but not TIME_FLAGS_PERMANENT)
<i>quantize</i>	A time object that holds a quantization interval, can be NULL for no quantization.

### 38.49.4.35 time\_schedule\_limit()

```
void time_schedule_limit (
    t_timeobject * x,
    t_timeobject * quantize )
```

Schedule a task, with optional minimum interval.,

#### Parameters

<i>x</i>	The time object that schedules temporary events (must have been created with TIME_FLAGS_USECLOCK but not TIME_FLAGS_PERMANENT)
<i>quantize</i>	The minimum interval into the future when the event can occur, can be NULL if there is no minimum interval.

### 38.49.4.36 time\_setclock()

```
void time_setclock (
    t_timeobject * tx,
    t_symbol * sc )
```

Associate a named setclock object with a time object (unsupported).

#### Parameters

<i>tx</i>	Time object.
<i>sc</i>	Name of an associated setclock object.

**38.49.4.37 time\_setvalue()**

```
void time_setvalue (
    t_timeobject * tx,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Set the current value of a time object (either an interval or a position) using a Max message.

**Parameters**

<i>tx</i>	The time object.
<i>s</i>	Message selector.
<i>argc</i>	Count of arguments.
<i>argv</i>	Message arguments.

**38.49.4.38 time\_stop()**

```
void time_stop (
    t_timeobject * x )
```

Stop a currently scheduled time object.

**Parameters**

<i>x</i>	The time object.
----------	------------------

**38.49.4.39 time\_tick()**

```
void time_tick (
    t_timeobject * x )
```

Execute a time object's task, then if it was already set to execute, reschedule for the current interval value of the object.

**Parameters**

<i>x</i>	The time object.
----------	------------------

### 38.49.5 Variable Documentation

#### 38.49.5.1 t\_timeobject

```
BEGIN_USING_C_LINKAGE typedef t_object t_timeobject
```

A high-level time object for tempo-based scheduling.

See also

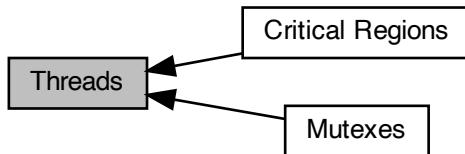
[t\\_itm](#)

[ITM](#)

## 38.50 Threads

In Max, there are several threads of execution.

Collaboration diagram for Threads:



## Modules

- [Critical Regions](#)

*A critical region is a simple mechanism that prevents multiple threads from accessing at once code protected by the same critical region.*

- [Mutexes](#)

## Macros

- `#define ATOMIC_INCREMENT(atomicptr)`  
*increment an atomic int value  
return value of ATOMIC\_INCREMENT and ATOMIC\_DECREMENT is the \*new\* value after performing the operation*
- `#define ATOMIC_INCREMENT_BARRIER(atomicptr)`  
*increment an atomic int value with a memory barrier  
return value of ATOMIC\_INCREMENT and ATOMIC\_DECREMENT is the \*new\* value after performing the operation*
- `#define ATOMIC_DECREMENT(atomicptr)`  
*decrement an atomic int value  
return value of ATOMIC\_INCREMENT and ATOMIC\_DECREMENT is the \*new\* value after performing the operation*
- `#define ATOMIC_DECREMENT_BARRIER(atomicptr)`  
*decrement an atomic int value with a memory barrier  
return value of ATOMIC\_INCREMENT and ATOMIC\_DECREMENT is the \*new\* value after performing the operation*

## Typedefs

- `typedef void * t_systhread_mutex`  
*An opaque mutex handle.*
- `typedef void * t_systhread_cond`  
*An opaque cond handle.*

## Enumerations

- `enum e_max_systhread_mutex_flags { SYSTHREAD_MUTEX_NORMAL , SYSTHREAD_MUTEX_ERRORCHECK , SYSTHREAD_MUTEX_RECURSIVE }`  
`systhread_mutex_new()` flags

## Functions

- `void schedule (void *ob, method fun, long when, t_symbol *sym, short argc, t_atom *argv)`  
*Cause a function to be executed at the timer level at some time in the future.*
- `void schedule_delay (void *ob, method fun, long delay, t_symbol *sym, short argc, t_atom *argv)`  
*Cause a function to be executed at the timer level at some time in the future specified by a delay offset.*
- `long isr (void)`  
*Determine whether your code is executing in the Max scheduler thread.*
- `void * defer (void *ob, method fn, t_symbol *sym, short argc, t_atom *argv)`  
*Defer execution of a function to the main thread if (and only if) your function is executing in the scheduler thread.*
- `void * defer_low (void *ob, method fn, t_symbol *sym, short argc, t_atom *argv)`  
*Defer execution of a function to the back of the queue on the main thread.*
- `long systhread_create (method entryproc, void *arg, long stacksize, long priority, long flags, t_systhread *thread)`  
*Create a new thread.*
- `long systhread_terminate (t_systhread thread)`  
*Forcefully kill a thread – not recommended.*
- `void systhread_sleep (long milliseconds)`  
*Suspend the execution of the calling thread.*

- void `systhread_exit` (long status)  
*Exit the calling thread.*
- long `systhread_join` (`t_systhread` thread, unsigned int \*retval)  
*Wait for thread to quit and get return value from `systhread_exit()`.*
- long `systhread_detach` (`t_systhread` thread)  
*Detach a thread.*
- `t_systhread systhread_self` (void)  
*Return the thread instance pointer for the calling thread.*
- long `systhread_equal` (`t_systhread` thread1, `t_systhread` thread2)  
*Compare two threads to see if they reference the same thread.*
- void `systhread_setpriority` (`t_systhread` thread, int priority)  
*Set the thread priority for the given thread.*
- int `systhread_getpriority` (`t_systhread` thread)  
*Get the thread priority for the given thread.*
- short `systhread_ismainthread` (void)  
*Check to see if the function currently being executed is in the main thread.*
- short `systhread_istimerthread` (void)  
*Check to see if the function currently being executed is in a scheduler thread.*
- short `systhread_isaudiothread` (void)  
*Check to see if the function currently being executed is in an audio thread.*
- void `systhread_set_name` (const char \*name)  
*Set the name of the current thread, for debugging purposes.*

## Variables

- `BEGIN_USING_C_LINKAGE` typedef void \* `t_systhread`  
*An opaque thread instance pointer.*

### 38.50.1 Detailed Description

In Max, there are several threads of execution.

The details of these threads are highlighted in the article "Event Priority in Max (Scheduler vs. Queue)" located online at <http://www.cycling74.com/story/2005/5/2/133649/9742>.

Not all of the details of Max's threading model are expounded here. Most important to understand is that we typically deal the scheduler (which when overdrive is on runs in a separate and high priority thread) and the low priority queue (which always runs in the main application thread).

#### See also

<http://www.cycling74.com/twiki/bin/view/ProductDocumentation/JitterSdk+SchedQueue>  
<http://www.cycling74.com/story/2005/5/2/133649/9742>

### 38.50.2 Enumeration Type Documentation

#### 38.50.2.1 e\_max\_systhread\_mutex\_flags

enum `e_max_systhread_mutex_flags`

`systhread_mutex_new()` flags

Enumerator

<code>SYSTHREAD_MUTEX_NORMAL</code>	Normal.
<code>SYSTHREAD_MUTEX_ERRORCHECK</code>	Error-checking.
<code>SYSTHREAD_MUTEX_RECURSIVE</code>	Recursive.

### 38.50.3 Function Documentation

#### 38.50.3.1 defer()

```
void* defer (
    void * ob,
    method fn,
    t_symbol * sym,
    short argc,
    t_atom * argv )
```

Defer execution of a function to the main thread if (and only if) your function is executing in the scheduler thread.

Parameters

<code>ob</code>	First argument passed to the function fun when it executes.
<code>fn</code>	Function to be called, see below for how it should be declared.
<code>sym</code>	Second argument passed to the function fun when it executes.
<code>argc</code>	Count of arguments in argv. argc is also the third argument passed to the function fun when it executes.
<code>argv</code>	Array containing a variable number of <code>t_atom</code> function arguments. If this argument is non-zero, defer allocates memory to make a copy of the arguments (according to the size passed in argc) and passes the copied array to the function fun when it executes as the fourth argument.

**Returns**

Return values is for internal Cycling '74 use only.

**Remarks**

This function uses the [isr\(\)](#) routine to determine whether you're at the Max timer interrupt level (in the scheduler thread). If so, [defer\(\)](#) creates a Qelem (see [Qelems](#)), calls [qelem\\_front\(\)](#), and its queue function calls the function fn you passed with the specified arguments. If you're not in the scheduler thread, the function is executed immediately with the arguments. Note that this implies that [defer\(\)](#) is not appropriate for using in situations such as Device or File manager I/O completion routines. The [defer\\_low\(\)](#) function is appropriate however, because it always defers.

The deferred function should be declared as follows:

```
void myobject_do (myObject *client, t_symbol *s, short argc, t_atom *argv);
```

**See also**

[defer\\_low\(\)](#)

Referenced by [jit\\_error\\_code\(\)](#), [jit\\_error\\_sym\(\)](#), and [jit\\_post\\_sym\(\)](#).

**38.50.3.2 defer\_low()**

```
void* defer_low (
    void * ob,
    method fn,
    t_symbol * sym,
    short argc,
    t_atom * argv )
```

Defer execution of a function to the back of the queue on the main thread.

**Parameters**

<i>ob</i>	First argument passed to the function fun when it executes.
<i>fn</i>	Function to be called, see below for how it should be declared.
<i>sym</i>	Second argument passed to the function fun when it executes.
<i>argc</i>	Count of arguments in argv. argc is also the third argument passed to the function fun when it executes.
<i>argv</i>	Array containing a variable number of <a href="#">t_atom</a> function arguments. If this argument is non-zero, defer allocates memory to make a copy of the arguments (according to the size passed in argc) and passes the copied array to the function fun when it executes as the fourth argument.

**Returns**

Return values is for internal Cycling '74 use only.

### Remarks

`defer_low()` always defers a call to the function `fun` whether you are already in the main thread or not, and uses `qelem_set()`, not `qelem_front()`. This function is recommended for responding to messages that will cause your object to open a dialog box, such as read and write.

The deferred function should be declared as follows:

```
void myobject_do (myObject *client, t_symbol *s, short argc, t_atom *argv);
```

### See also

[defer\(\)](#)

### 38.50.3.3 isr()

```
long isr (
    void )
```

Determine whether your code is executing in the Max scheduler thread.

### Returns

This function returns non-zero if you are within Max's scheduler thread, zero otherwise. Note that if your code sets up other types of interrupt-level callbacks, such as for other types of device drivers used in asynchronous mode, `isr` will return false.

### 38.50.3.4 schedule()

```
void schedule (
    void * ob,
    method fun,
    long when,
    t_symbol * sym,
    short argc,
    t_atom * argv )
```

Cause a function to be executed at the timer level at some time in the future.

### Parameters

<code>ob</code>	First argument passed to the function <code>fun</code> when it executes.
<code>fun</code>	Function to be called, see below for how it should be declared.
<code>when</code>	The logical time that the function <code>fun</code> will be executed.
<code>sym</code>	Second argument passed to the function <code>fun</code> when it executes.
<code>argc</code>	Count of arguments in <code>argv</code> . <code>argc</code> is also the third argument passed to the function <code>fun</code> when it executes.
<code>argv</code> <sup>v74</sup>	Array containing a variable number of <code>t_atom</code> function arguments. If this argument is non-zero, <code>defer</code> allocates memory to make a copy of the arguments (according to the size passed in <code>argc</code> ) and passes the copied array to the function <code>fun</code> when it executes as the fourth argument.

### Remarks

`schedule()` calls a function at some time in the future. Unlike `defer()`, the function is called in the scheduling loop when logical time is equal to the specified value when. This means that the function could be called at interrupt level, so it should follow the usual restrictions on interrupt-level conduct. The function fun passed to `schedule` should be declared as follows:

```
void myobject_do (myObject *client, t_symbol *s, short argc, t_atom *argv);
```

### Remarks

One use of `schedule()` is as an alternative to using the lockout flag.

### See also

[defer\(\)](#)

### 38.50.3.5 `schedule_delay()`

```
void schedule_delay (
    void * ob,
    method fun,
    long delay,
    t_symbol * sym,
    short argc,
    t_atom * argv )
```

Cause a function to be executed at the timer level at some time in the future specified by a delay offset.

### Parameters

<i>ob</i>	First argument passed to the function fun when it executes.
<i>fun</i>	Function to be called, see below for how it should be declared.
<i>delay</i>	The delay from the current time before the function will be executed.
<i>sym</i>	Second argument passed to the function fun when it executes.
<i>argc</i>	Count of arguments in argv. argc is also the third argument passed to the function fun when it executes.
<i>argv</i>	Array containing a variable number of <code>t_atom</code> function arguments. If this argument is non-zero, <code>schedule_delay()</code> allocates memory to make a copy of the arguments (according to the size passed in argc) and passes the copied array to the function fun when it executes as the fourth argument.

### Remarks

`schedule_delay()` is similar to `schedule()` but allows you to specify the time as a delay rather than a specific logical time.

```
void myobject_click (t_myobject *x, Point pt, short modifiers)
{
    t_atom a[1];
    a[0].a_type = A_LONG;
```

```

        a[0].a_w.w_long = Random();
        schedule_delay(x, myobject_sched, 0, 0, 1, a);
}
void myobject_sched (t_myobject **x, t_symbol *s, short ac, t_atom *av)
{
    outlet_int(x->m_out, av->a_w.w_long);
}

```

**See also**[schedule\(\)](#)**38.50.3.6 systhread\_create()**

```

long systhread_create (
    method entryproc,
    void * arg,
    long stacksize,
    long priority,
    long flags,
    t_systhread * thread )

```

Create a new thread.

**Parameters**

<i>entryproc</i>	A method to call in the new thread when the thread is created.
<i>arg</i>	An argument to pass to the method specified for entryproc. Typically this might be a pointer to your object's struct.
<i>stacksize</i>	Not used. Pass 0 for this argument.
<i>priority</i>	Pass 0 for default priority. The priority can range from -32 to 32 where -32 is low, 0 is default and 32 is high.
<i>flags</i>	Not used. Pass 0 for this argument.
<i>thread</i>	The address of a <a href="#">t_systhread</a> where this thread's instance pointer will be stored.

**Returns**

A Max error code as defined in [e\\_max\\_errorcodes](#).

**38.50.3.7 systhread\_detach()**

```

long systhread_detach (
    t_systhread thread )

```

Detach a thread.

After detaching a thread you cannot call [systhread\\_join\(\)](#) on it.

**Parameters**

<i>thread</i>	The thread to join.
---------------	---------------------

**Returns**

A Max error code as defined in [e\\_max\\_errorcodes](#).

**Remarks**

You should either call [systhread\\_join\(\)](#) on a thread or [systhread\\_detach\(\)](#) to allow the system to reclaim resources.

**38.50.3.8 systhread\_equal()**

```
long systhread_equal (
    t_systhread thread1,
    t_systhread thread2 )
```

Compare two threads to see if they reference the same thread.

The *t\_systhread* type is opaque and two should not be compared directly.

**Parameters**

<i>thread1</i>	the first thread to be compared
<i>thread2</i>	the second thread to be compared

**Returns**

nonzero if the two parameters reference the same thread

**38.50.3.9 systhread\_exit()**

```
void systhread_exit (
    long status )
```

Exit the calling thread.

Call this from within a thread made using [systhread\\_create\(\)](#) when the thread is no longer needed.

**Parameters**

<i>status</i>	You will typically pass 0 for status. This value will be accessible by <a href="#">systhread_join()</a> , if needed.
---------------	--

**38.50.3.10 systhread\_getpriority()**

```
int systhread_getpriority (
    t_systhread thread )
```

Get the thread priority for the given thread.

**Parameters**

<i>thread</i>	The thread for which to find the priority.
---------------	--

**Returns**

The current priority value for the given thread.

**38.50.3.11 systhread\_isaudiothread()**

```
short systhread_isaudiothread (
    void )
```

Check to see if the function currently being executed is in an audio thread.

**Returns**

Returns true if the function is being executed in an audio thread, otherwise false.

**38.50.3.12 systhread\_ismainthread()**

```
short systhread_ismainthread (
    void )
```

Check to see if the function currently being executed is in the main thread.

**Returns**

Returns true if the function is being executed in the main thread, otherwise false.

### 38.50.3.13 `systhread_istimerthread()`

```
short systhread_istimerthread (
    void )
```

Check to see if the function currently being executed is in a scheduler thread.

#### Returns

Returns true if the function is being executed in a scheduler thread, otherwise false.

### 38.50.3.14 `systhread_join()`

```
long systhread_join (
    t_systhread thread,
    unsigned int * retval )
```

Wait for thread to quit and get return value from [systhread\\_exit\(\)](#).

#### Parameters

<i>thread</i>	The thread to join.
<i>retval</i>	The address of a long to hold the return value (status) from <a href="#">systhread_exit()</a> .

#### Returns

A Max error code as defined in [e\\_max\\_errorcodes](#).

#### Remarks

If your object is freed, and your thread function accesses memory from your object, then you will obviously have a memory violation. A common use of [systhread\\_join\(\)](#) is to prevent this situation by waiting (in your free method) for the thread to exit.

### 38.50.3.15 `systhread_self()`

```
t_systhread systhread_self (
    void )
```

Return the thread instance pointer for the calling thread.

#### Returns

The thread instance pointer for the thread from which this function is called.

### 38.50.3.16 systhread\_set\_name()

```
void systhread_set_name (
    const char * name )
```

Set the name of the current thread, for debugging purposes.

Recommended to call from the top of the entryproc passed to systhread\_create

#### Parameters

<i>name</i>	The name to be given.
-------------	-----------------------

### 38.50.3.17 systhread\_setpriority()

```
void systhread_setpriority (
    t_systhread thread,
    int priority )
```

Set the thread priority for the given thread.

#### Parameters

<i>thread</i>	The thread for which to set the priority.
<i>priority</i>	A value in the range -32 to 32 where -32 is lowest, 0 is default, and 32 is highest.

### 38.50.3.18 systhread\_sleep()

```
void systhread_sleep (
    long milliseconds )
```

Suspend the execution of the calling thread.

#### Parameters

<i>milliseconds</i>	The number of milliseconds to suspend the execution of the calling thread. The actual amount of time may be longer depending on various factors.
---------------------	--

### 38.50.3.19 systhread\_terminate()

```
long systhread_terminate (
    t_systhread thread )
```

Forcefully kill a thread – not recommended.

#### Parameters

<i>thread</i>	The thread to kill.
---------------	---------------------

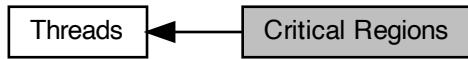
#### Returns

A Max error code as defined in [e\\_max\\_errorcodes](#).

## 38.51 Critical Regions

A critical region is a simple mechanism that prevents multiple threads from accessing at once code protected by the same critical region.

Collaboration diagram for Critical Regions:



## Functions

- void [critical\\_new \(t\\_critical \\*x\)](#)  
*Create a new critical region.*
- void [critical\\_enter \(t\\_critical x\)](#)  
*Enter a critical region.*
- void [critical\\_exit \(t\\_critical x\)](#)  
*Leave a critical region.*
- void [critical\\_free \(t\\_critical x\)](#)  
*Free a critical region created with [critical\\_new\(\)](#).*
- short [critical\\_tryenter \(t\\_critical x\)](#)  
*Try to enter a critical region if it is not locked.*

## Variables

- `BEGIN_USING_C_LINKAGE` `typedef pthread_mutex_t * t_critical`  
*a critical region*

### 38.51.1 Detailed Description

A critical region is a simple mechanism that prevents multiple threads from accessing at once code protected by the same critical region.

The code fragments could be different, and in completely different modules, but as long as the critical region is the same, no two threads should call the protected code at the same time. If one thread is inside a critical region, and another thread wants to execute code protected by the same critical region, the second thread must wait for the first thread to exit the critical region. In some implementations a critical region can be set so that if it takes too long for the first thread to exit said critical region, the second thread is allowed to execute, dangerously and potentially causing crashes. This is the case for the critical regions exposed by Max and the default upper limit for a given thread to remain inside a critical region is two seconds. Despite the fact that there are two seconds of leeway provided before two threads can dangerously enter a critical region, it is important to only protect as small a portion of code as necessary with a critical region.

Under Max 4.1 and earlier there was a simple protective mechanism called "lockout" that would prevent the scheduler from interrupting the low priority thread during sensitive operations such as sending data out an outlet or modifying members of a linked list. This lockout mechanism has been deprecated, and under the Mac OS X and Windows XP versions (Max 4.2 and later) does nothing. So how do you protect thread sensitive operations? Use critical regions (also known as critical sections). However, it is very important to mention that all outlet calls are now thread safe and should never be contained inside a critical region. Otherwise, this could result in serious timing problems. For other tasks which are not thread safe, such as accessing a linked list, critical regions or some other thread protection mechanism are appropriate.

In Max, the `critical_enter()` function is used to enter a critical region, and the `critical_exit()` function is used to exit a critical region. It is important that in any function which uses critical regions, all control paths protected by the critical region, exit the critical region (watch out for goto or return statements). The `critical_enter()` and `critical_exit()` functions take a critical region as an argument. However, for almost all purposes, we recommend using the global critical region in which case this argument is zero. The use of multiple critical regions can cause problems such as deadlock, i.e. when thread #1 is inside critical region A waiting on critical region B, but thread #2 is inside critical region B and is waiting on critical region A. In a flexible programming environment such as Max, deadlock conditions are easier to generate than you might think. So unless you are completely sure of what you are doing, and absolutely need to make use of multiple critical regions to protect your code, we suggest you use the global critical region.

In the following example code we show how one might use critical regions to protect the traversal of a linked list, testing to find the first element whose values is equal to "val". If this code were not protected, another thread which was modifying the linked list could invalidate assumptions in the traversal code.

```
critical_enter(0);
for (p = head; p; p = p->next) {
    if (p->value == val)
        break;
}
critical_exit(0);
return p;
```

And just to illustrate how to ensure a critical region is exited when multiple control paths are protected by a critical region, here's a slight variant.

```
critical_enter(0);
for (p = head; p; p = p->next) {
    if (p->value == val) {
        critical_exit(0);
```

```

        return p;
    }
critical_exit(0);
return NULL;
}

```

For more information on multi-threaded programming, hardware interrupts, and related topics, we suggest you perform some research online or read the relevant chapters of "Modern Operating Systems" by Andrew S. Tanenbaum (Prentice Hall). At the time of writing, some relevant chapters from this book are available for download in PDF format on Prentice Hall's web site. See:

[http://www.prenhall.com/divisions/esm/app/author\\_tanenbaum/custom/mos2e/](http://www.prenhall.com/divisions/esm/app/author_tanenbaum/custom/mos2e/)

Look under "sample sections".

## 38.51.2 Function Documentation

### 38.51.2.1 critical\_enter()

```
void critical_enter (
    t_critical x )
```

Enter a critical region.

Typically you will want the argument to be zero to enter the global critical region, although you could pass your own critical created with [critical\\_new\(\)](#). It is important to try to keep the amount of code in the critical region to a minimum. Exit the critical region with [critical\\_exit\(\)](#).

#### Parameters

x	A pointer to a <a href="#">t_critical</a> struct, or zero to uses Max's global critical region.
---	---

#### See also

[critical\\_exit\(\)](#)

Referenced by [jit\\_global\\_critical\\_enter\(\)](#).

### 38.51.2.2 critical\_exit()

```
void critical_exit (
    t_critical x )
```

Leave a critical region.

Typically you will want the argument to be zero to exit the global critical region, although, if you are using your own critical regions you will want to pass the same one that you previously passed to [critical\\_enter\(\)](#).

**Parameters**

x	A pointer to a <a href="#">t_critical</a> struct, or zero to uses Max's global critical region.
---	---

Referenced by `jit_global_critical_exit()`.

### 38.51.2.3 critical\_free()

```
void critical_free (
    t_critical x )
```

Free a critical region created with [critical\\_new\(\)](#).

If you created your own critical region, you will need to free it in your object's free method.

**Parameters**

x	The <a href="#">t_critical</a> struct that will be freed.
---	---

### 38.51.2.4 critical\_new()

```
void critical_new (
    t_critical * x )
```

Create a new critical region.

Normally, you do not need to create your own critical region, because you can use Max's global critical region. Only use this function (in your object's instance creation method) if you are certain you are not able to use the global critical region.

**Parameters**

x	A <a href="#">t_critical</a> struct will be returned to you via this pointer.
---	---

### 38.51.2.5 critical\_tryenter()

```
short critical_tryenter (
    t_critical x )
```

Try to enter a critical region if it is not locked.

**Parameters**

x	A pointer to a <a href="#">t_critical</a> struct, or zero to uses Max's global critical region.
---	---

**Returns**

returns non-zero if there was a problem entering

**See also**

[critical\\_enter\(\)](#)

## 38.52 Mutexes

Collaboration diagram for Mutexes:



## Functions

- long [systhread\\_mutex\\_new](#) ([t\\_systhread\\_mutex](#) \*pmutex, long flags)  
*Create a new mutex, which can be used to place thread locks around critical code.*
- long [systhread\\_mutex\\_free](#) ([t\\_systhread\\_mutex](#) pmutex)  
*Free a mutex created with [systhread\\_mutex\\_new\(\)](#).*
- long [systhread\\_mutex\\_lock](#) ([t\\_systhread\\_mutex](#) pmutex)  
*Enter block of locked code code until a [systhread\\_mutex\\_unlock\(\)](#) is reached.*
- long [systhread\\_mutex\\_unlock](#) ([t\\_systhread\\_mutex](#) pmutex)  
*Exit a block of code locked with [systhread\\_mutex\\_lock\(\)](#).*
- long [systhread\\_mutex\\_trylock](#) ([t\\_systhread\\_mutex](#) pmutex)  
*Try to enter block of locked code code until a [systhread\\_mutex\\_unlock\(\)](#) is reached.*
- long [systhread\\_mutex\\_newlock](#) ([t\\_systhread\\_mutex](#) \*pmutex, long flags)  
*Convenience utility that combines [systhread\\_mutex\\_new\(\)](#) and [systhread\\_mutex\\_lock\(\)](#).*

### 38.52.1 Detailed Description

**See also**

[Critical Regions](#)

## 38.52.2 Function Documentation

### 38.52.2.1 systhread\_mutex\_free()

```
long systhread_mutex_free (
    t_systhread_mutex *pmutex )
```

Free a mutex created with [systhread\\_mutex\\_new\(\)](#).

#### Parameters

<i>pmutex</i>	The mutex instance pointer.
---------------	-----------------------------

#### Returns

A Max error code as defined in [e\\_max\\_errorcodes](#).

### 38.52.2.2 systhread\_mutex\_lock()

```
long systhread_mutex_lock (
    t_systhread_mutex *pmutex )
```

Enter block of locked code code until a [systhread\\_mutex\\_unlock\(\)](#) is reached.

It is important to keep the code in this block as small as possible.

#### Parameters

<i>pmutex</i>	The mutex instance pointer.
---------------	-----------------------------

#### Returns

A Max error code as defined in [e\\_max\\_errorcodes](#).

#### See also

[systhread\\_mutex\\_trylock\(\)](#)

### 38.52.2.3 systhread\_mutex\_new()

```
long systhread_mutex_new (
    t_systhread_mutex * pmutex,
    long flags )
```

Create a new mutex, which can be used to place thread locks around critical code.

The mutex should be freed with [systhread\\_mutex\\_free\(\)](#).

#### Parameters

<i>pmutex</i>	The address of a variable to store the mutex pointer.
<i>flags</i>	Flags to determine the behaviour of the mutex, as defined in <a href="#">e_max_systhread_mutex_flags</a> .

#### Returns

A Max error code as defined in [e\\_max\\_errorcodes](#).

#### Remarks

One reason to use [systhread\\_mutex\\_new\(\)](#) instead of [Critical Regions](#) is to create non-recursive locks, which are lighter-weight than recursive locks.

### 38.52.2.4 systhread\_mutex\_newlock()

```
long systhread_mutex_newlock (
    t_systhread_mutex * pmutex,
    long flags )
```

Convenience utility that combines [systhread\\_mutex\\_new\(\)](#) and [systhread\\_mutex\\_lock\(\)](#).

#### Parameters

<i>pmutex</i>	The address of a variable to store the mutex pointer.
<i>flags</i>	Flags to determine the behaviour of the mutex, as defined in <a href="#">e_max_systhread_mutex_flags</a> .

#### Returns

A Max error code as defined in [e\\_max\\_errorcodes](#).

### 38.52.2.5 systhread\_mutex\_trylock()

```
long systhread_mutex_trylock (
    t_systhread_mutex *pmutex )
```

Try to enter block of locked code until a [systhread\\_mutex\\_unlock\(\)](#) is reached.

If the lock cannot be entered, this function will return non-zero.

#### Parameters

<i>pmutex</i>	The mutex instance pointer.
---------------	-----------------------------

#### Returns

Returns non-zero if there was a problem entering.

#### See also

[systhread\\_mutex\\_lock\(\)](#)

### 38.52.2.6 systhread\_mutex\_unlock()

```
long systhread_mutex_unlock (
    t_systhread_mutex *pmutex )
```

Exit a block of code locked with [systhread\\_mutex\\_lock\(\)](#).

#### Parameters

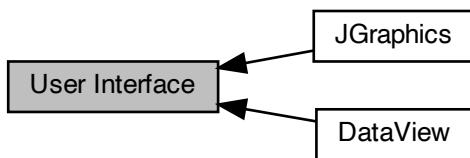
<i>pmutex</i>	The mutex instance pointer.
---------------	-----------------------------

**Returns**

A Max error code as defined in [e\\_max\\_errorcodes](#).

## 38.53 User Interface

Collaboration diagram for User Interface:



## Modules

- [JGraphics](#)

*JGraphics is the API for creating user interface objects introduced with Max 5.*

- [DataView](#)

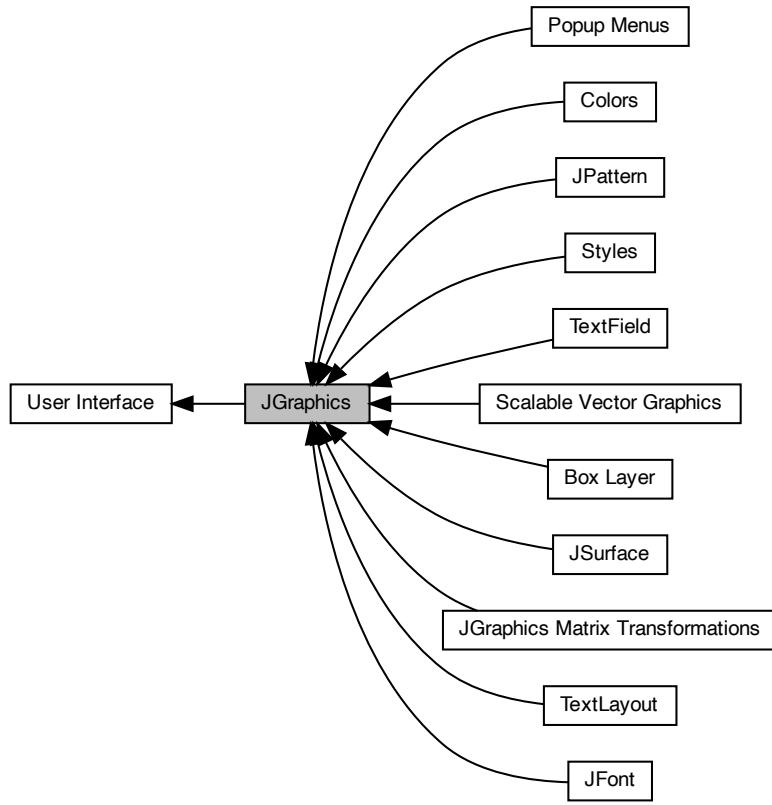
*The jdataview object provides a mechanism to display data in a tabular format.*

### 38.53.1 Detailed Description

## 38.54 JGraphics

JGraphics is the API for creating user interface objects introduced with Max 5.

Collaboration diagram for JGraphics:



## Modules

- [JSurface](#)

*A surface is an abstract base class for something you render to.*

- [Scalable Vector Graphics](#)

- [JFont](#)

- [JGraphics Matrix Transformations](#)

*The `t_jmatrix` is one way to represent a transformation.*

- [JPattern](#)

*A pattern is like a brush that is used to fill a path with.*

- [Colors](#)

- [Styles](#)

*Styles provide a means by which to inherit attribute values from a patcher that are consistently used across many objects.*

- [TextField](#)

*The textfield is a high-level text display object that may be used by a UI object to represent text in a patcher.*

- [TextLayout](#)

*A textlayout is lower-level text rendering object used by higher-level entities such as [TextField](#).*

- [Popup Menus](#)

*Popup menu API so externals can create popup menus that can also be drawn into.*

- [Box Layer](#)

*The boxlayer functions provide way to make it easier to use cached offscreen images (layers) in your drawing.*

## Data Structures

- struct [t\\_jgraphics\\_font\\_extents](#)

*A structure for holding information related to how much space the rendering of a given font will use.*

## Macros

- #define [JGRAPHICS\\_RECT\\_BOTTOM\(rect\)](#)

*Determine the coordinate of the bottom of a rect.*

- #define [JGRAPHICS\\_RECT\\_RIGHT\(rect\)](#)

*Determine the coordinate of the right side of a rect.*

- #define [JGRAPHICS\\_PI](#)

*Utility macro to return the value of Pi.*

- #define [JGRAPHICS\\_2PI](#)

*Utility macro to return the value of twice Pi.*

- #define [JGRAPHICS\\_PIOVER2](#)

*Utility macro to return the value of half of Pi.*

- #define [JGRAPHICS\\_3PIOVER2](#)

*Utility macro to return the 270° Case.*

## Typedefs

- typedef [typedefBEGIN\\_USING\\_C\\_LINKAGE](#) struct \_jgraphics [t\\_jgraphics](#)

*An instance of a jgraphics drawing context.*

- typedef struct \_jpath [t\\_jpath](#)

*An instance of a jgraphics path.*

- typedef struct \_jtextlayout [t\\_jtextlayout](#)

*An instance of a jgraphics text layout object.*

- typedef struct \_jtransform [t\\_jtransform](#)

*An instance of a jgraphics transform.*

- typedef struct \_jdesktopui [t\\_jdesktopui](#)

*An instance of a transparent UI window on the desktop.*

- typedef struct \_jpopupmenu [t\\_jpopupmenu](#)

*An instance of a pop-up menu.*

- typedef struct \_jsvg [t\\_jsvg](#)

*An instance of an SVG object.*

- typedef struct \_jsvg\_remap [t\\_jsvg\\_remap](#)

*An object used for remapping colors in a t\_svg.*

## Enumerations

- enum `t_jgraphics_format` { `JGRAPHICS_FORMAT_ARGB32`, `JGRAPHICS_FORMAT_RGB24`, `JGRAPHICS_FORMAT_A8` }

*Enumeration of color formats used by jgraphics surfaces.*

- enum `t_jgraphics_fileformat` { `JGRAPHICS_FILEFORMAT_PNG`, `JGRAPHICS_FILEFORMAT_JPEG` }

*Enumeration of file formats usable for jgraphics surfaces.*

- enum `t_jgraphics_text_justification` {  
`JGRAPHICS_TEXT JUSTIFICATION LEFT`, `JGRAPHICS_TEXT JUSTIFICATION RIGHT`, `JGRAPHICS_TEXT JUSTIFICATION CENTERED`,  
`JGRAPHICS_TEXT JUSTIFICATION TOP`,  
`JGRAPHICS_TEXT JUSTIFICATION BOTTOM`, `JGRAPHICS_TEXT JUSTIFICATION VCENTERED`,  
`JGRAPHICS_TEXT JUSTIFICATION HJUSTIFIED`, `JGRAPHICS_TEXT JUSTIFICATION CENTERED` }

*Enumeration of text justification options, which are specified as a bitmask.*

## Functions

- int `jgraphics_round` (double d)  
*Utility for rounding a double to an int.*
- `t_jgraphics * jgraphics_reference` (`t_jgraphics *g`)  
*Get a reference to a graphics context.*
- void `jgraphics_destroy` (`t_jgraphics *g`)  
*Release or free a graphics context.*
- void `jgraphics_new_path` (`t_jgraphics *g`)  
*Begin a new path.*
- `t_jpath * jgraphics_copy_path` (`t_jgraphics *g`)  
*Get a copy of the current path from a context.*
- `t_jpath * jgraphics_path_createstroked` (`t_jpath *p`, double thickness, `t_jgraphics_line_join` join, `t_jgraphics_line_cap` cap)  
*Create a new path consisting of the original path stroked with a given thickness.*
- void `jgraphics_path_destroy` (`t_jpath *path`)  
*Release/free a path.*
- void `jgraphics_append_path` (`t_jgraphics *g`, `t_jpath *path`)  
*Add a path to a graphics context.*
- void `jgraphics_close_path` (`t_jgraphics *g`)  
*Close the current path in a context.*
- void `jgraphics_path_roundcorners` (`t_jgraphics *g`, double cornerRadius)  
*Round out any corners in a path.*
- long `jgraphics_path_contains` (`t_jpath *path`, double x, double y)  
*Test if the path contains the point x,y.*
- long `jgraphics_path_intersectsline` (`t_jpath *path`, double x1, double y1, double x2, double y2)  
*Test if the path intersects the line defined by x1,y1 and x2,y2.*
- double `jgraphics_path_getlength` (`t_jpath *path`)  
*Return the length of a path.*
- void `jgraphics_path_getpointalongpath` (`t_jpath *path`, double distancefromstart, double \*x, double \*y)  
*Return a point that lies a given distance from the start of the path.*
- double `jgraphics_path_getnearestpoint` (`t_jpath *path`, double x, double y, double \*path\_x, double \*path\_y)  
*Finds the point on the path that is nearest to the point x,y passed in.*

- long `jgraphics_path_getpathelems` (`t_jpath` \*`path`, `t_jgraphics_path_elem` \*\*`elems`)  
*Get the path elements and return number of path elements.*
- void `jgraphics_get_current_point` (`t_jgraphics` \*`g`, double \*`x`, double \*`y`)  
*Get the current location of the cursor in a graphics context.*
- void `jgraphics_arc` (`t_jgraphics` \*`g`, double `xc`, double `yc`, double `radius`, double `angle1`, double `angle2`)  
*Add a circular, clockwise, arc to the current path.*
- void `jgraphics_ovalarc` (`t_jgraphics` \*`g`, double `xc`, double `yc`, double `radiusx`, double `radiusy`, double `angle1`, double `angle2`)  
*Add a non-circular arc to the current path.*
- void `jgraphics_arc_negative` (`t_jgraphics` \*`g`, double `xc`, double `yc`, double `radius`, double `angle1`, double `angle2`)  
*Add a circular, counter-clockwise, arc to the current path.*
- void `jgraphics_curve_to` (`t_jgraphics` \*`g`, double `x1`, double `y1`, double `x2`, double `y2`, double `x3`, double `y3`)  
*Add a cubic Bezier spline to the current path.*
- void `jgraphics_rel_curve_to` (`t_jgraphics` \*`g`, double `x1`, double `y1`, double `x2`, double `y2`, double `x3`, double `y3`)  
*Add a cubic Bezier spline to the current path, using coordinates relative to the current point.*
- void `jgraphics_line_to` (`t_jgraphics` \*`g`, double `x`, double `y`)  
*Add a line segment to the current path.*
- void `jgraphics_rel_line_to` (`t_jgraphics` \*`g`, double `x`, double `y`)  
*Add a line segment to the current path, using coordinates relative to the current point.*
- void `jgraphics_move_to` (`t_jgraphics` \*`g`, double `x`, double `y`)  
*Move the cursor to a new point and begin a new subpath.*
- void `jgraphics_rel_move_to` (`t_jgraphics` \*`g`, double `x`, double `y`)  
*Move the cursor to a new point and begin a new subpath, using coordinates relative to the current point.*
- void `jgraphics_rectangle` (`t_jgraphics` \*`g`, double `x`, double `y`, double `width`, double `height`)  
*Add a closed rectangle path in the context.*
- void `jgraphics_oval` (`t_jgraphics` \*`g`, double `x`, double `y`, double `width`, double `height`)  
*Deprecated – do not use.*
- void `jgraphics_rectangle_rounded` (`t_jgraphics` \*`g`, double `x`, double `y`, double `width`, double `height`, double `ovalwidth`, double `ovalheight`)  
*Add a closed rounded-rectangle path in the context.*
- void `jgraphics_ellipse` (`t_jgraphics` \*`g`, double `x`, double `y`, double `width`, double `height`)  
*Add a closed elliptical path in the context.*
- void `jgraphics_bubble` (`t_jgraphics` \*`g`, double `bodyx`, double `bodyy`, double `bodywidth`, double `bodyheight`, double `cornersize`, double `arrowtipx`, double `arrowtipy`, `t_jgraphics_bubble_side` `whichside`, double `arrowedgeprop`, double `arrowwidth`)  
*Add a closed bubble path in the context.*
- void `jgraphics_triangle` (`t_jgraphics` \*`g`, double `x1`, double `y1`, double `x2`, double `y2`, double `x3`, double `y3`)  
*Add a closed triangular path in the context.*
- void `jgraphics_select_font_face` (`t_jgraphics` \*`g`, const char \*`family`, `t_jgraphics_font_slant` `slant`, `t_jgraphics_font_weight` `weight`)  
*Specify a font for a graphics context.*
- void `jgraphics_select_font` (`t_jgraphics` \*`g`, `t_jfont` \*`jfont`)  
*Specify a font for a graphics context by passing a `t_jfont` object.*
- void `jgraphics_set_font_size` (`t_jgraphics` \*`g`, double `size`)  
*Specify the font size for a context.*
- void `jgraphics_set_underline` (`t_jgraphics` \*`g`, char `underline`)  
*Turn underlining on/off for text in a context.*
- void `jgraphics_show_text` (`t_jgraphics` \*`g`, const char \*`utf8`)

- void `jgraphics_text_path` (`t_jgraphics` \*g, const char \*utf8)
 

*Add a path of text to the current path.*
- void `jgraphics_font_extents` (`t_jgraphics` \*g, `t_jgraphics_font_extents` \*extents)
 

*Return the extents of the currently selected font for a given graphics context.*
- void `jgraphics_text_measure` (`t_jgraphics` \*g, const char \*utf8, double \*width, double \*height)
 

*Return the height and width of a string given current graphics settings in a context.*
- void `jgraphics_text_measuretext_wrapped` (`t_jgraphics` \*g, const char \*utf8, double wrapwidth, long includewhitespace, double \*width, double \*height, long \*numlines)
 

*Return the height, width, and number of lines that will be used to render a given string.*
- long `jgraphics_system_canantialiastexttotransparentbg` ()
 

*Determine if you can anti-alias text to a transparent background.*
- void `jgraphics_user_to_device` (`t_jgraphics` \*g, double \*x, double \*y)
 

*User coordinates are those passed to drawing functions in a given `t_jgraphics` context.*
- void `jgraphics_device_to_user` (`t_jgraphics` \*g, double \*x, double \*y)
 

*User coordinates are those passed to drawing functions in a given `t_jgraphics` context.*
- void `jgraphics_getfiletypes` (void \*dummy, long \*count, `t_fourcc` \*\*filetypes, char \*alloc)
 

*Get a list of filetypes appropriate for use with jgraphics surfaces.*
- long `jgraphics_rectintersectsrect` (`t_rect` \*r1, `t_rect` \*r2)
 

*Simple utility to test for rectangle intersection.*
- long `jgraphics_rectcontainsrect` (`t_rect` \*outer, `t_rect` \*inner)
 

*Simple utility to test for rectangle containment.*
- void `jgraphics_position_one_rect_near_another_rect_but_keep_inside_a_third_rect` (`t_rect` \*positioned\_rect, const `t_rect` \*positioned\_near\_this\_rect, const `t_rect` \*keep\_inside\_this\_rect)
 

*Generate a `t_rect` according to positioning rules.*
- void `jgraphics_clip` (`t_jgraphics` \*g, double x, double y, double width, double height)
 

*Clip to a subset of the graphics context; once done, cannot be undone, only further reduced.*

### 38.54.1 Detailed Description

JGraphics is the API for creating user interface objects introduced with Max 5.

It includes functions for drawing vector-based shapes, managing pop-up menus, rendering text, and importing graphics resources. The API design is inspired by and analogous to the `Cairo` API, though the underlying implementation is actually drawn using `JUCE` (JUCE functions, however, cannot be called directly).

### 38.54.2 Macro Definition Documentation

#### 38.54.2.1 JGRAPHICS\_2PI

```
#define JGRAPHICS_2PI
```

Utility macro to return the value of twice Pi.

### 38.54.2.2 JGRAPHICS\_3PIOVER2

```
#define JGRAPHICS_3PIOVER2
```

Utility macro to return the 270° Case.

### 38.54.2.3 JGRAPHICS\_PI

```
#define JGRAPHICS_PI
```

Utility macro to return the value of Pi.

### 38.54.2.4 JGRAPHICS\_PIOVER2

```
#define JGRAPHICS_PIOVER2
```

Utility macro to return the value of half of Pi.

## 38.54.3 Enumeration Type Documentation

### 38.54.3.1 t\_jgraphics\_fileformat

```
enum t_jgraphics_fileformat
```

Enumeration of file formats usable for jgraphics surfaces.

Enumerator

JGRAPHICS_FILEFORMAT_PNG	Portable Network Graphics (PNG) format.
JGRAPHICS_FILEFORMAT_JPEG	JPEG format.

### 38.54.3.2 t\_jgraphics\_format

```
enum t_jgraphics_format
```

Enumeration of color formats used by jgraphics surfaces.

**Enumerator**

JGRAPHICS_FORMAT_ARGB32	Color is represented using 32 bits, 8 bits each for the components, and including an alpha component.
JGRAPHICS_FORMAT_RGB24	Color is represented using 32 bits, 8 bits each for the components. There is no alpha component.
JGRAPHICS_FORMAT_A8	The color is represented only as an 8-bit alpha mask.

**38.54.3.3 t\_jgraphics\_text\_justification**

```
enum t_jgraphics_text_justification
```

Enumeration of text justification options, which are specified as a bitmask.

**Enumerator**

JGRAPHICS_TEXT JUSTIFICATION LEFT	Justify left.
JGRAPHICS_TEXT JUSTIFICATION RIGHT	Justify right.
JGRAPHICS_TEXT JUSTIFICATION HCENTERED	Centered horizontally.
JGRAPHICS_TEXT JUSTIFICATION TOP	Justified to the top.
JGRAPHICS_TEXT JUSTIFICATION BOTTOM	Justified to the bottom.
JGRAPHICS_TEXT JUSTIFICATION VCENTERED	Centered vertically.
JGRAPHICS_TEXT JUSTIFICATION HJUSTIFIED	Horizontally justified.
JGRAPHICS_TEXT JUSTIFICATION CENTERED	Shortcut for Centering both vertically and horizontally.

**38.54.4 Function Documentation****38.54.4.1 jgraphics\_append\_path()**

```
void jgraphics_append_path (
    t_jgraphics * g,
    t_jpath * path )
```

Add a path to a graphics context.

**Parameters**

<i>g</i>	The graphics context.
<i>path</i>	The path to add.

#### 38.54.4.2 jgraphics\_arc()

```
void jgraphics_arc (
    t_jgraphics * g,
    double xc,
    double yc,
    double radius,
    double angle1,
    double angle2 )
```

Add a circular, clockwise, arc to the current path.

##### Parameters

<i>g</i>	The graphics context.
<i>xc</i>	The horizontal coordinate of the arc's center.
<i>yc</i>	The vertical coordinate of the arc's center.
<i>radius</i>	The radius of the arc.
<i>angle1</i>	The starting angle of the arc in radians. Zero radians is center right (positive x axis).
<i>angle2</i>	The terminal angle of the arc in radians. Zero radians is center right (positive x axis).

#### 38.54.4.3 jgraphics\_arc\_negative()

```
void jgraphics_arc_negative (
    t_jgraphics * g,
    double xc,
    double yc,
    double radius,
    double angle1,
    double angle2 )
```

Add a circular, counter-clockwise, arc to the current path.

##### Parameters

<i>g</i>	The graphics context.
<i>xc</i>	The horizontal coordinate of the arc's center.
<i>yc</i>	The vertical coordinate of the arc's center.
<i>radius</i>	The radius of the arc.
<i>angle1</i>	The starting angle of the arc in radians. Zero radians is center right (positive x axis).
<i>angle2</i>	The terminal angle of the arc in radians. Zero radians is center right (positive x axis).

### 38.54.4.4 jgraphics\_bubble()

```
void jgraphics_bubble (
    t_jgraphics * g,
    double bodyx,
    double bodyy,
    double bodywidth,
    double bodyheight,
    double cornersize,
    double arrowtipx,
    double arrowtipy,
    t_jgraphics_bubble_side whichside,
    double arrowedgeprop,
    double arrowwidth )
```

Add a closed bubble path in the context.

#### Parameters

<i>g</i>	The graphics context.
<i>bodyx</i>	Horizontal body origin.
<i>bodyy</i>	The vertical origin.
<i>bodywidth</i>	The width of the rect.
<i>bodyheight</i>	The height of the rect.
<i>cornersize</i>	Body rounded corners
<i>arrowtipx</i>	X position of arrow tip
<i>arrowtipy</i>	Y position of arrow tip
<i>whichside</i>	side to connect arrow, see above definition of <i>t_jgraphics_bubble_side</i> enum,
<i>arrowedgeprop</i>	Arrow proportion along edge (0-1)
<i>arrowwidth</i>	Arrow base width

### 38.54.4.5 jgraphics\_clip()

```
void jgraphics_clip (
    t_jgraphics * g,
    double x,
    double y,
    double width,
    double height )
```

Clip to a subset of the graphics context; once done, cannot be undone, only further reduced.

**Parameters**

<i>g</i>	The <code>t_jgraphics</code> context to be clipped.
<i>x</i>	x origin of clip region.
<i>y</i>	y origin of clip region.
<i>width</i>	width of clip region.
<i>height</i>	height of clip region.

**38.54.4.6 `jgraphics_close_path()`**

```
void jgraphics_close_path (
    t_jgraphics * g )
```

Close the current path in a context.

This will add a line segment to close current subpath.

**Parameters**

<i>g</i>	The graphics context.
----------	-----------------------

**38.54.4.7 `jgraphics_copy_path()`**

```
t_jpath* jgraphics_copy_path (
    t_jgraphics * g )
```

Get a copy of the current path from a context.

**Parameters**

<i>g</i>	the graphics context containing the current path
----------	--

**Returns**

A copy of the current path.

### 38.54.4.8 jgraphics\_curve\_to()

```
void jgraphics_curve_to (
    t_jgraphics * g,
    double x1,
    double y1,
    double x2,
    double y2,
    double x3,
    double y3 )
```

Add a cubic Bezier spline to the current path.

#### Parameters

<i>g</i>	The graphics context.
<i>x1</i>	The first control point.
<i>y1</i>	The first control point.
<i>x2</i>	The second control point.
<i>y2</i>	The second control point.
<i>x3</i>	The destination point.
<i>y3</i>	The destination point.

### 38.54.4.9 jgraphics\_destroy()

```
void jgraphics_destroy (
    t_jgraphics * g )
```

Release or free a graphics context.

#### Parameters

<i>g</i>	The context to release.
----------	-------------------------

### 38.54.4.10 jgraphics\_device\_to\_user()

```
void jgraphics_device_to_user (
    t_jgraphics * g,
    double * x,
    double * y )
```

User coordinates are those passed to drawing functions in a given [t\\_jgraphics](#) context.

Device coordinates refer to patcher canvas coordinates, before any zooming.

#### 38.54.4.11 jgraphics\_ellipse()

```
void jgraphics_ellipse (
    t_jgraphics * g,
    double x,
    double y,
    double width,
    double height )
```

Add a closed elliptical path in the context.

##### Parameters

<i>g</i>	The graphics context.
<i>x</i>	The horizontal origin.
<i>y</i>	The vertical origin.
<i>width</i>	The width of the rect.
<i>height</i>	The height of the rect.

#### 38.54.4.12 jgraphics\_font\_extents()

```
void jgraphics_font_extents (
    t_jgraphics * g,
    t_jgraphics_font_extents * extents )
```

Return the extents of the currently selected font for a given graphics context.

##### Parameters

<i>g</i>	Pointer to a jgraphics context.
<i>extents</i>	The address of a <a href="#">t_jgraphics_font_extents</a> structure to be filled with the results.

#### 38.54.4.13 jgraphics\_get\_current\_point()

```
void jgraphics_get_current_point (
    t_jgraphics * g,
    double * x,
    double * y )
```

Get the current location of the cursor in a graphics context.

**Parameters**

<i>g</i>	The graphics context.
<i>x</i>	The address of a variable that will be set to the horizontal cursor location upon return.
<i>y</i>	The address of a variable that will be set to the vertical cursor location upon return.

**38.54.4.14 jgraphics\_getfiletypes()**

```
void jgraphics_getfiletypes (
    void * dummy,
    long * count,
    t_fourcc ** filetypes,
    char * alloc )
```

Get a list of of filetypes appropriate for use with jgraphics surfaces.

**Parameters**

<i>dummy</i>	Unused.
<i>count</i>	The address of a variable to be set with the number of types in filetypes upon return.
<i>filetypes</i>	The address of a variable that will represent the array of file types upon return.
<i>alloc</i>	The address of a char that will be flagged with a 1 or a 0 depending on whether or not memory was allocated for the filetypes member.

**Remarks**

This example shows a common usage of [jgraphics\\_getfiletypes\(\)](#).

```
char      filename[MAX_PATH_CHARS];
t_fourcc *type = NULL;
long      ntype;
long      outtype;
t_max_err err;
char      alloc;
short     path;
t_jsurface *surface;

if (want_to_show_dialog) {
    jgraphics_getfiletypes(x, &ntype, &type, &alloc);
    err = open_dialog(filename, &path, (void *)&outtype, (void *)type, ntype);
    if (err)
        goto out;
}
else {
    strncpy_zero(filename, s->s_name, MAX_PATH_CHARS);
    err = locatefile_extended(filename, &path, &outtype, type, ntype);
    if (err)
        goto out;
}
surface = jgraphics_image_surface_create_referenced(filename, path);
out:
if (alloc)
    sysmem_freeptr((char *)type);
```

#### 38.54.4.15 jgraphics\_line\_to()

```
void jgraphics_line_to (
    t_jgraphics * g,
    double x,
    double y )
```

Add a line segment to the current path.

##### Parameters

<i>g</i>	The graphics context.
<i>x</i>	The destination point.
<i>y</i>	The destination point.

#### 38.54.4.16 jgraphics\_move\_to()

```
void jgraphics_move_to (
    t_jgraphics * g,
    double x,
    double y )
```

Move the cursor to a new point and begin a new subpath.

##### Parameters

<i>g</i>	The graphics context.
<i>x</i>	The new location.
<i>y</i>	The new location.

#### 38.54.4.17 jgraphics\_new\_path()

```
void jgraphics_new_path (
    t_jgraphics * g )
```

Begin a new path.

This action clears any current path in the context.

##### Parameters

<i>g</i>	The graphics context.
----------	-----------------------

### 38.54.4.18 jgraphics\_oval()

```
void jgraphics_oval (
    t_jgraphics * g,
    double x,
    double y,
    double width,
    double height )
```

Deprecated – do not use.

Adds a closed oval path in the context, however, it does not scale appropriately.

#### Parameters

<i>g</i>	The graphics context.
<i>x</i>	The horizontal origin.
<i>y</i>	The vertical origin.
<i>width</i>	The width of the oval.
<i>height</i>	The height of the oval.

### 38.54.4.19 jgraphics\_ovalarc()

```
void jgraphics_ovalarc (
    t_jgraphics * g,
    double xc,
    double yc,
    double radiusx,
    double radiusy,
    double angle1,
    double angle2 )
```

Add a non-circular arc to the current path.

#### Parameters

<i>g</i>	The graphics context.
<i>xc</i>	The horizontal coordinate of the arc's center.
<i>yc</i>	The vertical coordinate of the arc's center.
<i>radiusx</i>	The horizontal radius of the arc.
<i>radiusy</i>	The vertical radius of the arc.
<i>angle1</i>	The starting angle of the arc in radians. Zero radians is center right (positive x axis).
<i>angle2</i>	The terminal angle of the arc in radians. Zero radians is center right (positive x axis).

#### 38.54.4.20 jgraphics\_path\_contains()

```
long jgraphics_path_contains (
    t_jpath * path,
    double x,
    double y )
```

Test if the path contains the point x,y.

##### Parameters

<i>path</i>	the path
<i>x</i>	the x-coordinate of the point to test
<i>y</i>	the y-coordinate of the point to test

#### 38.54.4.21 jgraphics\_path\_createstroked()

```
t_jpath* jgraphics_path_createstroked (
    t_jpath * p,
    double thickness,
    t_jgraphics_line_join join,
    t_jgraphics_line_cap cap )
```

Create a new path consisting of the original path stroked with a given thickness.

##### Parameters

<i>p</i>	the path to be stroked
<i>thickness</i>	thickness of the stroke
<i>join</i>	the style to join segments together at corners
<i>cap</i>	the style of end cap to use

##### Returns

the new path, which must be freed with [jgraphics\\_path\\_destroy\(\)](#) when done

#### 38.54.4.22 jgraphics\_path\_destroy()

```
void jgraphics_path_destroy (
    t_jpath * path )
```

Release/free a path.

**Parameters**

<i>path</i>	The path to release.
-------------	----------------------

**38.54.4.23 jgraphics\_path\_getlength()**

```
double jgraphics_path_getlength (
    t_jpath * path )
```

Return the length of a path.

**Parameters**

<i>path</i>	the path
-------------	----------

**Returns**

the length of the path

**38.54.4.24 jgraphics\_path\_getnearestpoint()**

```
double jgraphics_path_getnearestpoint (
    t_jpath * path,
    double x,
    double y,
    double * path_x,
    double * path_y )
```

Finds the point on the path that is nearest to the point x,y passed in.

**Parameters**

<i>path</i>	the path to search
<i>x</i>	x position of the target point
<i>y</i>	y position of the target point
<i>path_x</i>	pointer to double to receive the x position of closest point on path
<i>path_y</i>	pointer to double to receive the y position of the closest point on path

**Returns**

returns the distance along the path from the path start position to the found point on the path

**38.54.4.25 jgraphics\_path\_getpathelems()**

```
long jgraphics_path_getpathelems (
    t_jpath * path,
    t_jgraphics_path_elem ** elems )
```

Get the path elements and return number of path elements.

**Parameters**

<i>path</i>	the path
<i>elems</i>	pointer to array of path elements

**Returns**

the number of path elements

**38.54.4.26 jgraphics\_path\_getpointalongpath()**

```
void jgraphics_path_getpointalongpath (
    t_jpath * path,
    double distancefromstart,
    double * x,
    double * y )
```

Return a point that lies a given distance from the start of the path.

**Parameters**

<i>path</i>	the path
<i>distancefromstart</i>	distance from the start point
<i>x</i>	pointer to double to receive the x position of the point
<i>y</i>	pointer to double to receive the y position of the point

#### 38.54.4.27 jgraphics\_path\_intersectsline()

```
long jgraphics_path_intersectsline (
    t_jpath * path,
    double x1,
    double y1,
    double x2,
    double y2 )
```

Test if the path intersects the line defined by x1,y1 and x2,y2.

##### Parameters

<i>path</i>	the path
<i>x1</i>	the x-coordinate of the first point on the line
<i>y1</i>	the y-coordinate of the first point on the line
<i>x2</i>	the x-coordinate of the second point on the line
<i>y2</i>	the y-coordinate of the second point on the line

#### 38.54.4.28 jgraphics\_path\_roundcorners()

```
void jgraphics_path_roundcorners (
    t_jgraphics * g,
    double cornerRadius )
```

Round out any corners in a path.

This action clears any current path in the context.

##### Parameters

<i>g</i>	The graphics context.
<i>cornerRadius</i>	The amount by which to round corners.

#### 38.54.4.29 jgraphics\_position\_one\_rect\_near\_another\_rect\_but\_keep\_inside\_a\_third\_rect()

```
void jgraphics_position_one_rect_near_another_rect_but_keep_inside_a_third_rect (
    t_rect * positioned_rect,
    const t_rect * positioned_near_this_rect,
    const t_rect * keep_inside_this_rect )
```

Generate a [t\\_rect](#) according to positioning rules.

**Parameters**

<i>positioned_rect</i>	The address of a valid <code>t_rect</code> whose members will be filled in upon return.
<i>positioned_near_this_rect</i>	A pointer to a rect near which this rect should be positioned.
<i>keep_inside_this_rect</i>	A pointer to a rect defining the limits within which the new rect must reside.

**38.54.4.30 jgraphics\_rectangle()**

```
void jgraphics_rectangle (
    t_jgraphics * g,
    double x,
    double y,
    double width,
    double height )
```

Add a closed rectangle path in the context.

**Parameters**

<i>g</i>	The graphics context.
<i>x</i>	The horizontal origin.
<i>y</i>	The vertical origin.
<i>width</i>	The width of the rect.
<i>height</i>	The height of the rect.

**38.54.4.31 jgraphics\_rectangle\_rounded()**

```
void jgraphics_rectangle_rounded (
    t_jgraphics * g,
    double x,
    double y,
    double width,
    double height,
    double ovalwidth,
    double ovalheight )
```

Add a closed rounded-rectangle path in the context.

**Parameters**

<i>g</i>	The graphics context.
<i>x</i>	The horizontal origin.

**Parameters**

<i>y</i>	The vertical origin.
<i>width</i>	The width of the rect.
<i>height</i>	The height of the rect.
<i>ovalwidth</i>	The width of the oval used for the round corners.
<i>ovalheight</i>	The height of the oval used for the round corners.

**38.54.4.32 jgraphics\_rectcontainsrect()**

```
long jgraphics_rectcontainsrect (
    t_rect * outer,
    t_rect * inner )
```

Simple utility to test for rectangle containment.

**Parameters**

<i>outer</i>	The address of the first rect for the test.
<i>inner</i>	The address of the second rect for the test.

**Returns**

Returns true if the inner rect is completely inside the outer rect, otherwise false.

**38.54.4.33 jgraphics\_rectintersectsrect()**

```
long jgraphics_rectintersectsrect (
    t_rect * r1,
    t_rect * r2 )
```

Simple utility to test for rectangle intersection.

**Parameters**

<i>r1</i>	The address of the first rect for the test.
<i>r2</i>	The address of the second rect for the test.

**Returns**

Returns true if the rects intersect, otherwise false.

**38.54.4.34 jgraphics\_reference()**

```
t_jgraphics* jgraphics_reference (
    t_jgraphics * g )
```

Get a reference to a graphics context.

When you are done you should release your reference with [jgraphics\\_destroy\(\)](#).

**Parameters**

<i>g</i>	The context you wish to reference.
----------	------------------------------------

**Returns**

A new reference to the context.

**38.54.4.35 jgraphics\_rel\_curve\_to()**

```
void jgraphics_rel_curve_to (
    t_jgraphics * g,
    double x1,
    double y1,
    double x2,
    double y2,
    double x3,
    double y3 )
```

Add a cubic Bezier spline to the current path, using coordinates relative to the current point.

**Parameters**

<i>g</i>	The graphics context.
<i>x1</i>	The first control point.
<i>y1</i>	The first control point.
<i>x2</i>	The second control point.
<i>y2</i>	The second control point.
<i>x3</i>	The destination point.
<i>y3</i>	The destination point.

#### 38.54.4.36 jgraphics\_rel\_line\_to()

```
void jgraphics_rel_line_to (
    t_jgraphics * g,
    double x,
    double y )
```

Add a line segment to the current path, using coordinates relative to the current point.

##### Parameters

<i>g</i>	The graphics context.
<i>x</i>	The destination point.
<i>y</i>	The destination point.

#### 38.54.4.37 jgraphics\_rel\_move\_to()

```
void jgraphics_rel_move_to (
    t_jgraphics * g,
    double x,
    double y )
```

Move the cursor to a new point and begin a new subpath, using coordinates relative to the current point.

##### Parameters

<i>g</i>	The graphics context.
<i>x</i>	The new location.
<i>y</i>	The new location.

#### 38.54.4.38 jgraphics\_round()

```
int jgraphics_round (
    double d )
```

Utility for rounding a double to an int.

**Parameters**

<i>d</i>	floating-point input.
----------	-----------------------

**Returns**

rounded int output.

**38.54.4.39 jgraphics\_select\_font\_face()**

```
void jgraphics_select_font_face (
    t_jgraphics * g,
    const char * family,
    t_jgraphics_font_slant slant,
    t_jgraphics_font_weight weight )
```

Specify a font for a graphics context.

**Parameters**

<i>g</i>	The graphics context.
<i>family</i>	The name of the font family (e.g. "Arial").
<i>slant</i>	Define the slant to use for the font.
<i>weight</i>	Define the weight to use for the font.

**38.54.4.40 jgraphics\_select\_jfont()**

```
void jgraphics_select_jfont (
    t_jgraphics * g,
    t_jfont * jfont )
```

Specify a font for a graphics context by passing a [t\\_jfont](#) object.

**Parameters**

<i>g</i>	The graphics context.
<i>jfont</i>	A jfont object whose attributes will be copied to the context.

**38.54.4.41 jgraphics\_set\_font\_size()**

```
void jgraphics_set_font_size (
    t_jgraphics * g,
    double size )
```

Specify the font size for a context.

**Parameters**

<i>g</i>	The graphics context.
<i>size</i>	The font size.

**38.54.4.42 jgraphics\_set\_underline()**

```
void jgraphics_set_underline (
    t_jgraphics * g,
    char underline )
```

Turn underlining on/off for text in a context.

**Parameters**

<i>g</i>	The graphics context.
<i>underline</i>	Pass true or false to set the appropriate behavior.

**38.54.4.43 jgraphics\_show\_text()**

```
void jgraphics_show_text (
    t_jgraphics * g,
    const char * utf8 )
```

Display text at the current position in a context.

**Parameters**

<i>g</i>	The graphics context.
<i>utf8</i>	The text to display.

#### 38.54.4.44 jgraphics\_system\_canantialiastexttotransparentbg()

```
long jgraphics_system_canantialiastexttotransparentbg ( )
```

Determine if you can anti-alias text to a transparent background.

You might want to call this and then disable "useimagebuffer" if false \*and\* you are rendering text on a transparent background.

##### Returns

Non-zero if you can anti-alias text to a transparent background.

#### 38.54.4.45 jgraphics\_text\_measure()

```
void jgraphics_text_measure (
    t_jgraphics * g,
    const char * utf8,
    double * width,
    double * height )
```

Return the height and width of a string given current graphics settings in a context.

##### Parameters

<i>g</i>	Pointer to a jgraphics context.
<i>utf8</i>	A string containing the text whose dimensions we wish to find.
<i>width</i>	The address of a variable to be filled with the width of the rendered text.
<i>height</i>	The address of a variable to be filled with the height of the rendered text.

#### 38.54.4.46 jgraphics\_text\_measuretext\_wrapped()

```
void jgraphics_text_measuretext_wrapped (
    t_jgraphics * g,
    const char * utf8,
    double wrapwidth,
    long includewhitespace,
    double * width,
    double * height,
    long * numlines )
```

Return the height, width, and number of lines that will be used to render a given string.

**Parameters**

<i>g</i>	Pointer to a jgraphics context.
<i>utf8</i>	A string containing the text whose dimensions we wish to find.
<i>wrapwidth</i>	The number of pixels in width at which the text should be wrapped if it is too long.
<i>includewhitespace</i>	Set zero to not include white space in the calculation, otherwise set this parameter to 1.
<i>width</i>	The address of a variable to be filled with the width of the rendered text.
<i>height</i>	The address of a variable to be filled with the height of the rendered text.
<i>numlines</i>	The address of a variable to be filled with the number of lines required to render the text.

**38.54.4.47 jgraphics\_text\_path()**

```
void jgraphics_text_path (
    t_jgraphics * g,
    const char * utf8 )
```

Add a path of text to the current path.

**Parameters**

<i>g</i>	The graphics context.
<i>utf8</i>	The text to generate path for.

**38.54.4.48 jgraphics\_triangle()**

```
void jgraphics_triangle (
    t_jgraphics * g,
    double x1,
    double y1,
    double x2,
    double y2,
    double x3,
    double y3 )
```

Add a closed triangular path in the context.

**Parameters**

<i>g</i>	The graphics context.
<i>x1</i>	Coordinate for the first point.
<i>y1</i>	Coordinate for the first point.
<i>x2</i>	Coordinate for the second point.

### Parameters

<code>y2</code>	Coordinate for the second point.
<code>x3</code>	Coordinate for the third point.
<code>y3</code>	Coordinate for the third point.

### 38.54.4.49 `jgraphics_user_to_device()`

```
void jgraphics_user_to_device (
    t_jgraphics * g,
    double * x,
    double * y )
```

User coordinates are those passed to drawing functions in a given `t_jgraphics` context.

Device coordinates refer to patcher canvas coordinates, before any zooming.

## 38.55 JSurface

A surface is an abstract base class for something you render to.

Collaboration diagram for JSurface:



### Typedefs

- `typedef struct _jsurface t_jsurface`  
*An instance of a jgraphics surface.*

## Functions

- `t_jsurface * jgraphics_image_surface_create (t_jgraphics_format format, int width, int height)`  
*Create an image surface.*
- `t_jsurface * jgraphics_image_surface_create_referenced (const char *filename, short path)`  
*Create an image surface, filling it with the contents of a file, and get a reference to the surface.*
- `t_jsurface * jgraphics_image_surface_create_from_file (const char *filename, short path)`  
*Create an image surface, filling it with the contents of a file.*
- `t_jsurface * jgraphics_image_surface_create_for_data (unsigned char *data, t_jgraphics_format format, int width, int height, int stride, method freefun, void *freearg)`  
*Create an image surface from given pixel data.*
- `t_jsurface * jgraphics_image_surface_create_from_filedata (const void *data, unsigned long datalen)`  
*Create a new surface from file data.*
- `t_jsurface * jgraphics_image_surface_create_from_resource (const void *moduleRef, const char *resname)`  
*Create a new surface from a resource in your external.*
- `t_max_err jgraphics_get_resource_data (const void *moduleRef, const char *resname, long extcount, t_atom *exts, void **data, unsigned long *datasize)`  
*Low-level routine to access an object's resource data.*
- `t_jsurface * jgraphics_surface_reference (t_jsurface *s)`  
*Create a reference to an existing surface.*
- `void jgraphics_surface_destroy (t_jsurface *s)`  
*Release or free a surface.*
- `t_max_err jgraphics_image_surface_writepng (t_jsurface *surface, const char *filename, short path, long dpi)`  
*Export a PNG file of the contents of a surface.*
- `t_max_err jgraphics_image_surface_writejpeg (t_jsurface *surface, const char *filename, short path)`  
*Export a JPEG file of the contents of a surface.*
- `int jgraphics_image_surface_get_width (t_jsurface *s)`  
*Retrieve the width of a surface.*
- `int jgraphics_image_surface_get_height (t_jsurface *s)`  
*Retrieve the height of a surface.*
- `void jgraphics_image_surface_set_pixel (t_jsurface *s, int x, int y, t_jrgba color)`  
*Set the color of an individual pixel in a surface.*
- `void jgraphics_image_surface_get_pixel (t_jsurface *s, int x, int y, t_jrgba *color)`  
*Retrieve the color of an individual pixel in a surface.*
- `void jgraphics_image_surface_scroll (t_jsurface *s, int x, int y, int width, int height, int dx, int dy, t_jpath **path)`
- `void jgraphics_image_surface_draw (t_jgraphics *g, t_jsurface *s, t_rect srcRect, t_rect destRect)`  
*Draw an image surface.*
- `void jgraphics_image_surface_draw_fast (t_jgraphics *g, t_jsurface *s)`  
*Draw an image surface quickly.*
- `void jgraphics_write_image_surface_to_filedata (t_jsurface *surf, long fmt, void **data, long *size)`  
*Get surface data ready for manually writing to a file.*
- `void jgraphics_image_surface_clear (t_jsurface *s, int x, int y, int width, int height)`  
*Set all pixels in rect to 0.*
- `t_jgraphics * jgraphics_create (t_jsurface *target)`  
*Create a context to draw on a particular surface.*

### 38.55.1 Detailed Description

A surface is an abstract base class for something you render to.

An image surface is a concrete instance that renders to an image in memory, essentially an offscreen bitmap.

### 38.55.2 Function Documentation

#### 38.55.2.1 jgraphics\_create()

```
t_jgraphics* jgraphics_create (
    t_jsurface * target )
```

Create a context to draw on a particular surface.

When you are done, call [jgraphics\\_destroy\(\)](#).

##### Parameters

<i>target</i>	The surface to which to draw.
---------------	-------------------------------

##### Returns

The new graphics context.

#### 38.55.2.2 jgraphics\_get\_resource\_data()

```
t_max_err jgraphics_get_resource_data (
    const void * moduleRef,
    const char * resname,
    long extcount,
    t_atom * exts,
    void ** data,
    unsigned long * datasize )
```

Low-level routine to access an object's resource data.

##### Parameters

<i>moduleRef</i>	A pointer to your external's module, which is passed to your external's main() function when the class is loaded.
------------------	---

**Parameters**

<i>resname</i>	Base name of the resource data (without an extension)
<i>extcount</i>	Count of possible extensions (ignored on Windows)
<i>exts</i>	Array of symbol atoms containing possible filename extensions (ignored on Windows)
<i>data</i>	Returned resource data assigned to a pointer you supply
<i>datasize</i>	Size of the data returned

**Remarks**

You are responsible for freeing any data returned in the data pointer

**Returns**

A Max error code.

**38.55.2.3 jgraphics\_image\_surface\_clear()**

```
void jgraphics_image_surface_clear (
    t_jsurface * s,
    int x,
    int y,
    int width,
    int height )
```

Set all pixels in rect to 0.

**Parameters**

<i>s</i>	The surface to clear.
<i>x</i>	The horizontal origin of the rect to clear.
<i>y</i>	The vertical origin of the rect to clear.
<i>width</i>	The width of the rect to clear.
<i>height</i>	The height of the rect to clear.

**38.55.2.4 jgraphics\_image\_surface\_create()**

```
t_jsurface* jgraphics_image_surface_create (
    t_jgraphics_format format,
    int width,
    int height )
```

Create an image surface.

Use [jgraphics\\_surface\\_destroy\(\)](#) to free it when you are done.

#### Parameters

<i>format</i>	Defines the color format for the new surface.
<i>width</i>	Defines the width of the new surface.
<i>height</i>	Defines the height of the new surface.

#### Returns

A pointer to the new surface.

### 38.55.2.5 jgraphics\_image\_surface\_create\_for\_data()

```
t_jsurface* jgraphics_image_surface_create_for_data (
    unsigned char * data,
    t_jgraphics_format format,
    int width,
    int height,
    int stride,
    method freefun,
    void * freearg )
```

Create an image surface from given pixel data.

Data should point to start of top line of bitmap, stride tells how to get to next line. For upside down windows bitmaps, data = (pBits-(height-1)\*stride) and stride is a negative number.

#### Parameters

<i>data</i>	The data. For example, an RGBA image loaded in memory.
<i>format</i>	The format of the data.
<i>width</i>	The width of the new surface.
<i>height</i>	The height of the new surface.
<i>stride</i>	The number of bytes between the start of rows in the dat buffer.
<i>freefun</i>	If not NULL, freefun will be called when the surface is destroyed
<i>freearg</i>	This will be passed to freefun if/when freefun is called.

#### Returns

A pointer to the new surface.

### 38.55.2.6 `jgraphics_image_surface_create_from_file()`

```
t_jsurface* jgraphics_image_surface_create_from_file (
    const char * filename,
    short path )
```

Create an image surface, filling it with the contents of a file.

Use [jgraphics\\_surface\\_destroy\(\)](#) to free it when you are done.

#### Parameters

<i>filename</i>	The name of the file.
<i>path</i>	The path id of the file.

#### Returns

A pointer to the new surface.

### 38.55.2.7 `jgraphics_image_surface_create_from_filedata()`

```
t_jsurface* jgraphics_image_surface_create_from_filedata (
    const void * data,
    unsigned long datalen )
```

Create a new surface from file data.

#### Parameters

<i>data</i>	A pointer to the raw PNG or JPG bits.
<i>datalen</i>	The number of bytes in data.

#### Returns

The new surface.

#### See also

[jgraphics\\_write\\_image\\_surface\\_to\\_filedata\(\)](#)

### 38.55.2.8 `jgraphics_image_surface_create_from_resource()`

```
t_jsurface* jgraphics_image_surface_create_from_resource (
    const void * moduleRef,
    const char * resname )
```

Create a new surface from a resource in your external.

#### Parameters

<code>moduleRef</code>	A pointer to your external's module, which is passed to your external's main() function when the class is loaded.
<code>resname</code>	The name of the resource in the external.

#### Remarks

The following example shows an example of how this might be used in an external.

```
static s_my_surface = NULL;
int main(void *moduleRef)
{
    // (Do typical class initialization here)

    // now create the surface from a resource that we added to the Xcode/VisualStudio project
    s_my_surface = jgraphics_image_surface_create_from_resource(moduleRef, "myCoolImage");
    return 0;
}
```

### 38.55.2.9 `jgraphics_image_surface_create_referenced()`

```
t_jsurface* jgraphics_image_surface_create_referenced (
    const char * filename,
    short path )
```

Create an image surface, filling it with the contents of a file, and get a reference to the surface.

Use `jgraphics_surface_destroy()` to release your reference to the surface when you are done.

#### Parameters

<code>filename</code>	The name of the file.
<code>path</code>	The path id of the file.

#### Returns

A pointer to the new surface.

### 38.55.2.10 jgraphics\_image\_surface\_draw()

```
void jgraphics_image_surface_draw (
    t_jgraphics * g,
    t_jsurface * s,
    t_rect srcRect,
    t_rect destRect )
```

Draw an image surface.

This is not in cairo, but, it seems silly to have to make a brush to just draw an image. This doesn't support rotations, however.

#### Parameters

<i>g</i>	The graphics context in which to draw the surface.
<i>s</i>	The surface to draw.
<i>srcRect</i>	The rect within the surface that should be drawn.
<i>destRect</i>	The rect in the context to which to draw the <i>srcRect</i> .

#### See also

[jgraphics\\_image\\_surface\\_draw\\_fast\(\)](#)

### 38.55.2.11 jgraphics\_image\_surface\_draw\_fast()

```
void jgraphics_image_surface_draw_fast (
    t_jgraphics * g,
    t_jsurface * s )
```

Draw an image surface quickly.

The `draw_fast` version won't scale based on zoom factor or user transforms so make sure that this is what you want! Draws entire image, origin *\*can\** be shifted via zoom and user transforms (even though image is not scaled based on those same transforms)

#### Parameters

<i>g</i>	The graphics context in which to draw the surface.
<i>s</i>	The surface to draw.

#### See also

[jgraphics\\_image\\_surface\\_draw](#)

### 38.55.2.12 `jgraphics_image_surface_get_height()`

```
int jgraphics_image_surface_get_height (
    t_jsurface * s )
```

Retrieve the height of a surface.

#### Parameters

<code>s</code>	The surface to query.
----------------	-----------------------

#### Returns

The height of the surface.

### 38.55.2.13 `jgraphics_image_surface_get_pixel()`

```
void jgraphics_image_surface_get_pixel (
    t_jsurface * s,
    int x,
    int y,
    t_jrgba * color )
```

Retrieve the color of an individual pixel in a surface.

#### Parameters

<code>s</code>	The surface.
<code>x</code>	The horizontal coordinate of the pixel.
<code>y</code>	The vertical coordinate of the pixel.
<code>color</code>	The address of a valid <code>t_jrgba</code> struct whose values will be filled in with the color of the pixel upon return.

### 38.55.2.14 `jgraphics_image_surface_get_width()`

```
int jgraphics_image_surface_get_width (
    t_jsurface * s )
```

Retrieve the width of a surface.

#### Parameters

<code>s</code>	The surface to query.
----------------	-----------------------

**Returns**

The width of the surface.

**38.55.2.15 jgraphics\_image\_surface\_scroll()**

```
void jgraphics_image_surface_scroll (
    t_jsurface * s,
    int x,
    int y,
    int width,
    int height,
    int dx,
    int dy,
    t_jpath ** path )
```

**Parameters**

<i>s</i>	The surface to scroll.
<i>x</i>	The origin of the rect to scroll.
<i>y</i>	The origin of the rect to scroll.
<i>width</i>	The width of the rect to scroll.
<i>height</i>	The height of the rect to scroll.
<i>dx</i>	The amount to scroll the surface horizontally.
<i>dy</i>	The amount to scroll the surface vertically.
<i>path</i>	Can pass NULL if you are not interested in this info. Otherwise pass a pointer and it will be returned with a path containing the invalid region.

**38.55.2.16 jgraphics\_image\_surface\_set\_pixel()**

```
void jgraphics_image_surface_set_pixel (
    t_jsurface * s,
    int x,
    int y,
    t_jrgba color )
```

Set the color of an individual pixel in a surface.

**Parameters**

<i>s</i>	The surface.
<i>x</i>	The horizontal coordinate of the pixel.
<i>y</i>	The vertical coordinate of the pixel.
<i>color</i>	The color of the pixel.

### 38.55.2.17 jgraphics\_image\_surface\_writejpeg()

```
t_max_err jgraphics_image_surface_writejpeg (
    t_jsurface * surface,
    const char * filename,
    short path )
```

Export a JPEG file of the contents of a surface.

#### Parameters

<i>surface</i>	The surface to export.
<i>filename</i>	Specify the name of the file to create.
<i>path</i>	Specify the path id for where to create the file.

#### Returns

A Max error code.

### 38.55.2.18 jgraphics\_image\_surface\_writepng()

```
t_max_err jgraphics_image_surface_writepng (
    t_jsurface * surface,
    const char * filename,
    short path,
    long dpi )
```

Export a PNG file of the contents of a surface.

#### Parameters

<i>surface</i>	The surface to export.
<i>filename</i>	Specify the name of the file to create.
<i>path</i>	Specify the path id for where to create the file.
<i>dpi</i>	Define the resolution of the image (e.g. 72).

#### Returns

A Max error code.

### 38.55.2.19 jgraphics\_surface\_destroy()

```
void jgraphics_surface_destroy (
    t_jsurface * s )
```

Release or free a surface.

#### Parameters

<i>s</i>	The surface to release.
----------	-------------------------

### 38.55.2.20 jgraphics\_surface\_reference()

```
t_jsurface* jgraphics_surface_reference (
    t_jsurface * s )
```

Create a reference to an existing surface.

Use [jgraphics\\_surface\\_destroy\(\)](#) to release your reference to the surface when you are done.

#### Parameters

<i>s</i>	The surface to reference.
----------	---------------------------

#### Returns

The new reference to the surface.

### 38.55.2.21 jgraphics\_write\_image\_surface\_to\_filedata()

```
void jgraphics_write_image_surface_to_filedata (
    t_jsurface * surf,
    long fmt,
    void ** data,
    long * size )
```

Get surface data ready for manually writing to a file.

#### Parameters

<i>surf</i>	The surface whose data will be retrieved.
<i>fmt</i>	The format for the data. This should be a selection from <a href="#">t_jgraphics_fileformat</a> .
<i>data</i>	The address of a pointer that will be allocated and filled. When you are done with this data you should free it using <a href="#">sysmem_freeptr()</a> .
<small>Cycling '74</small> <i>size</i>	The address of a variable to hold the size of the data upon return.

## Remarks

A good example of this is to embed the surface as a PNG in a patcher file.

```
long size = 0;
void *data = NULL;
jgraphics_write_image_surface_to_filedata(x->j_surface, JGRAPHICS_FILEFORMAT_PNG, &data, &size);
if (size) {
    x->j_format = gensym("png");
    binarydata_appendtodictionary(data, size, gensym("data"), x->j_format, d);
    x->j_imagedata = data;
    x->j_imagedatasize = size;
}
```

## See also

[jgraphics\\_image\\_surface\\_create\\_from\\_filedata\(\)](#)

## 38.56 Scalable Vector Graphics

Collaboration diagram for Scalable Vector Graphics:



## Functions

- [`t\_jsvg \* jsvg\_create\_from\_file`](#) (const char \*filename, short path)  
*Read an SVG file, return a `t_jsvg` object.*
- [`t\_jsvg \* jsvg\_create\_from\_resource`](#) (const void \*moduleRef, const char \*resname)  
*Read an SVG file from a resource.*
- [`t\_jsvg \* jsvg\_create\_from\_xmlstring`](#) (const char \*svgXML)  
*Create an SVG object from a string containing the SVG's XML.*
- [`void jsvg\_get\_size \(t\_jsvg \*svg, double \*width, double \*height\)`](#)  
*Retrieve the size of an SVG object.*
- [`void jsvg\_destroy \(t\_jsvg \*svg\)`](#)  
*Free a `t_jsvg` object.*
- [`void jsvg\_render \(t\_jsvg \*svg, t\_jgraphics \*g\)`](#)  
*Render an SVG into a graphics context.*

### 38.56.1 Detailed Description

### 38.56.2 Function Documentation

### 38.56.2.1 `jsvg_create_from_file()`

```
t_jsvg* jsvg_create_from_file (
    const char * filename,
    short path )
```

Read an SVG file, return a `t_jsvg` object.

#### Parameters

<code>filename</code>	The name of the file to read.
<code>path</code>	The path id of the file to read.

#### Returns

A new SVG object.

### 38.56.2.2 `jsvg_create_from_resource()`

```
t_jsvg* jsvg_create_from_resource (
    const void * moduleRef,
    const char * resname )
```

Read an SVG file from a resource.

#### Parameters

<code>moduleRef</code>	The external's moduleRef.
<code>resname</code>	The name of the SVG resource.

#### Returns

A new SVG object.

#### See also

[jgraphics\\_image\\_surface\\_create\\_from\\_resource\(\)](#)

### 38.56.2.3 `jsvg_create_from_xmlstring()`

```
t_jsvg* jsvg_create_from_xmlstring (
    const char * svgXML )
```

Create an SVG object from a string containing the SVG's XML.

**Parameters**

<code>svgXML</code>	The SVG source.
---------------------	-----------------

**Returns**

A new SVG object.

**38.56.2.4 `jsvg_destroy()`**

```
void jsvg_destroy (
    t_jsvg * svg )
```

Free a `t_jsvg` object.

**Parameters**

<code>svg</code>	The object to free.
------------------	---------------------

**38.56.2.5 `jsvg_get_size()`**

```
void jsvg_get_size (
    t_jsvg * svg,
    double * width,
    double * height )
```

Retrieve the size of an SVG object.

**Parameters**

<code>svg</code>	An SVG object.
<code>width</code>	The address of a variable that will be set to the width upon return.
<code>height</code>	The address of a variable that will be set to the width upon return.

**38.56.2.6 `jsvg_render()`**

```
void jsvg_render (
    t_jsvg * svg,
    t_jgraphics * g )
```

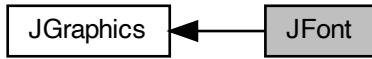
Render an SVG into a graphics context.

#### Parameters

<code>svg</code>	The SVG object to render.
<code>g</code>	The graphics context in which to render.

## 38.57 JFont

Collaboration diagram for JFont:



### Typedefs

- `typedef struct _jfont t_jfont`  
*An instance of a jgraphics font.*

### Enumerations

- `enum t_jgraphics_font_slant { JGRAPHICS_FONT_SLANT_NORMAL , JGRAPHICS_FONT_SLANT_ITALIC }`  
*Enumeration of slanting options for font display.*
- `enum t_jgraphics_font_weight{ JGRAPHICS_FONT_WEIGHT_NORMAL , JGRAPHICS_FONT_WEIGHT_BOLD }`  
*Enumeration of font weight options for font display.*

### Functions

- `t_jfont * jfont_create (const char *family, t_jgraphics_font_slant slant, t_jgraphics_font_weight weight, double size)`  
*Create a new font object.*
- `t_jfont * jfont_create_withstylename (const char *family, const char *stylename, double size)`  
*Create a new font object using a style instead of style flags.*
- `t_jfont * jfont_reference (t_jfont *font)`  
*Create new reference to an existing font object.*
- `void jfont_destroy (t_jfont *font)`  
*Release or free a font object.*

- long `jfont_isequalto (t_jfont *font, t_jfont *other)`  
*Compare two fonts to see if they are equivalent.*
- void `jfont_set_family (t_jfont *font, t_symbol *family)`  
*Set the name of the font family (e.g.*
- `t_symbol * jfont_get_family (t_jfont *font)`  
*Get the name of the font family (e.g.*
- void `jfont_set_style (t_jfont *font, t_symbol *style)`  
*Set the style of a font family (e.g.*
- `t_symbol * jfont_get_style (t_jfont *font)`  
*Get the style of the given font (e.g.*
- void `jfont_set_slant (t_jfont *font, t_jgraphics_font_slant slant)`  
*Set the slant of the font.*
- `t_jgraphics_font_slant jfont_get_slant (t_jfont *font)`  
*Get the slant of the font.*
- void `jfont_set_weight (t_jfont *font, t_jgraphics_font_weight weight)`  
*Set the weight of the font.*
- `t_jgraphics_font_weight jfont_get_weight (t_jfont *font)`  
*Get the weight of the font.*
- void `jfont_set_font_size (t_jfont *font, double size)`  
*Set the size of a font object.*
- double `jfont_get_font_size (t_jfont *font)`  
*Get the size of a font object.*
- void `jfont_set_underline (t_jfont *font, char ul)`  
*Set the underlining of a font object.*
- char `jfont_get_underline (t_jfont *font)`  
*Get the underline state of a font object.*
- void `jfont_extents (t_jfont *font, t_jgraphics_font_extents *extents)`  
*Get extents of this font.*
- void `jfont_text_measure (t_jfont *font, const char *utf8, double *width, double *height)`  
*Given a font, find out how much area is required to render a string of text.*
- void `jfont_text_measuretext_wrapped (t_jfont *font, const char *utf8, double wrapwidth, long includewhitespace, double *width, double *height, long *numlines)`  
*Given a font, find out how much area is required to render a string of text, provided a horizontal maximum limit at which the text is wrapped.*
- void `jfont_get_em_dimensions (t_jfont *font, double *width, double *height)`  
*Given a font, find out the width and height of the 'M' character.*
- `t_max_err jfont_getfontlist (long *count, t_symbol ***list)`  
*Get a list of font names.*
- `t_max_err jfont_getfontstylenames (t_symbol *fontname, long *count, t_symbol ***stylenames)`  
*Get a list of font styles available for a given font face.*
- void `jfont_normalizefontname (t_symbol *fontfacewithstyle, t_symbol **fontface, t_symbol **style)`  
*Given a font name with style appended separate into the component parts.*
- long `jbox_get_font_weight (t_object *b)`  
*Get the slant box's font.*
- long `jbox_get_font_slant (t_object *b)`  
*Get the slant box's font.*
- const char \* `systemfontname (void)`

*Retrieve the name of Max's system font.*

- const char \* [systemfontname\\_bold](#) (void)

*Retrieve the name of Max's bold system font.*

- const char \* [systemfontname\\_light](#) (void)

*Retrieve the name of Max's light system font.*

- [t\\_symbol \\* systemfontsym](#) (void)

*Retrieve the name of Max's system font as a symbol.*

### 38.57.1 Detailed Description

### 38.57.2 Enumeration Type Documentation

#### 38.57.2.1 [t\\_jgraphics\\_font\\_slant](#)

enum [t\\_jgraphics\\_font\\_slant](#)

Enumeration of slanting options for font display.

Enumerator

<a href="#">JGRAPHICS_FONT_SLANT_NORMAL</a>	Normal slanting (typically this means no slanting)
<a href="#">JGRAPHICS_FONT_SLANT_ITALIC</a>	Italic slanting.

#### 38.57.2.2 [t\\_jgraphics\\_font\\_weight](#)

enum [t\\_jgraphics\\_font\\_weight](#)

Enumeration of font weight options for font display.

Enumerator

<a href="#">JGRAPHICS_FONT_WEIGHT_NORMAL</a>	Normal font weight.
<a href="#">JGRAPHICS_FONT_WEIGHT_BOLD</a>	Bold font weight.

### 38.57.3 Function Documentation

### 38.57.3.1 jbox\_get\_font\_slant()

```
long jbox_get_font_slant (
    t_object * b )
```

Get the slant box's font.

#### Parameters

<i>b</i>	An object's box.
----------	------------------

#### Returns

A value from the [t\\_jgraphics\\_font\\_slant](#) enum.

### 38.57.3.2 jbox\_get\_font\_weight()

```
long jbox_get_font_weight (
    t_object * b )
```

Get the slant box's font.

#### Parameters

<i>b</i>	An object's box.
----------	------------------

#### Returns

A value from the [t\\_jgraphics\\_font\\_weight](#) enum.

### 38.57.3.3 jfont\_create()

```
t_jfont* jfont_create (
    const char * family,
    t_jgraphics_font_slant slant,
    t_jgraphics_font_weight weight,
    double size )
```

Create a new font object.

**Parameters**

<i>family</i>	The name of the font family (e.g. Arial).
<i>slant</i>	The type of slant for the font.
<i>weight</i>	The type of weight for the font.
<i>size</i>	The size of the font.

**Returns**

The new font object. Needs to be freed via [jfont\\_destroy\(\)](#) when done.

**38.57.3.4 jfont\_create\_withstylename()**

```
t_jfont* jfont_create_withstylename (
    const char * family,
    const char * stylename,
    double size )
```

Create a new font object using a style instead of style flags.

**Parameters**

<i>family</i>	The name of the font family (e.g. Arial).
<i>stylename</i>	The name of the font style (e.g. Narrow Bold Italic).
<i>size</i>	The size of the font.

**Returns**

The new font object. Needs to be freed via [jfont\\_destroy\(\)](#) when done.

**38.57.3.5 jfont\_destroy()**

```
void jfont_destroy (
    t_jfont * font )
```

Release or free a font object.

**Parameters**

<i>font</i>	The font object to release.
-------------	-----------------------------

### 38.57.3.6 jfont\_extents()

```
void jfont_extents (
    t_jfont * font,
    t_jgraphics_font_extents * extents )
```

Get extents of this font.

#### Parameters

<i>font</i>	The font object.
<i>extents</i>	The font extents upon return/

### 38.57.3.7 jfont\_get\_em\_dimensions()

```
void jfont_get_em_dimensions (
    t_jfont * font,
    double * width,
    double * height )
```

Given a font, find out the width and height of the 'M' character.

This is equivalent to jfont\_text\_measure(font, "M", width, height) but is faster.

#### Parameters

<i>font</i>	The font object.
<i>width</i>	The address of a variable to hold the width upon return.
<i>height</i>	The address of a variable to hold the height upon return.

### 38.57.3.8 jfont\_get\_family()

```
t_symbol* jfont_get_family (
    t_jfont * font )
```

Get the name of the font family (e.g.

Arial).

**Parameters**

<i>font</i>	The font object.
-------------	------------------

**Returns**

A [t\\_symbol](#) representing the name of the font family.

**38.57.3.9 jfont\_get\_font\_size()**

```
double jfont_get_font_size (
    t_jfont * font )
```

Get the size of a font object.

**Parameters**

<i>font</i>	The font object.
-------------	------------------

**Returns**

The size of the font.

**38.57.3.10 jfont\_get\_slant()**

```
t_jgraphics_font_slant jfont_get_slant (
    t_jfont * font )
```

Get the slant of the font.

**Parameters**

<i>font</i>	The font object.
-------------	------------------

**Returns**

The current slant setting for the font.

### 38.57.3.11 jfont\_get\_style()

```
t_symbol* jfont_get_style (
    t_jfont * font )
```

Get the style of the given font (e.g.

Narrow Bold Italic).

#### Parameters

<i>font</i>	The font object.
-------------	------------------

#### Returns

A `t_symbol` representing the name of the font style.

### 38.57.3.12 jfont\_get\_underline()

```
char jfont_get_underline (
    t_jfont * font )
```

Get the underline state of a font object.

#### Parameters

<i>font</i>	The font object.
-------------	------------------

#### Returns

Nonzero value if the font will be underlined.

### 38.57.3.13 jfont\_get\_weight()

```
t_jgraphics_font_weight jfont_get_weight (
    t_jfont * font )
```

Get the weight of the font.

**Parameters**

<i>font</i>	The font object.
-------------	------------------

**Returns**

The current weight setting for the font.

**38.57.3.14 jfont\_getfontlist()**

```
t_max_err jfont_getfontlist (
    long * count,
    t_symbol *** list )
```

Get a list of font names.

Note, this includes each font style in each font family.

**Parameters**

<i>count</i>	The address of a variable to hold the count of font names in list upon return.
<i>list</i>	The address of a <i>t_symbol</i> ** initialized to NULL. Upon return this will be set to an array of <i>count t_symbol</i> pointers. This array should be freed using <i>sysmem_freeptr()</i> when you are done with it.

**Returns**

A Max error code.

**38.57.3.15 jfont\_getfontstylenames()**

```
t_max_err jfont_getfontstylenames (
    t_symbol * fontname,
    long * count,
    t_symbol *** stylenames )
```

Get a list of font styles available for a given font face.

**Parameters**

<i>fontname</i>	A symbol with the name of the font whose styles are to be retrieved.
<i>count</i>	The address of a variable to hold the count of font styles in the list upon return.
<i>list</i>	The address of a <i>t_symbol</i> ** initialized to NULL. Upon return this will be set to an array of <i>count t_symbol</i> pointers. This array should be freed using <i>sysmem_freeptr()</i> when you are done with it.

**Returns**

A Max error code.

**38.57.3.16 jfont\_isequalto()**

```
long jfont_isequalto (
    t_jfont * font,
    t_jfont * other )
```

Compare two fonts to see if they are equivalent.

**Parameters**

<i>font</i>	The first font object that is being compared.
<i>other</i>	The second font object that is being compared.

**Returns**

Nonzero value if the two fonts are equivalent.

**38.57.3.17 jfont\_normalizefontname()**

```
void jfont_normalizefontname (
    t_symbol * fontfacewithstyle,
    t_symbol ** fontface,
    t_symbol ** style )
```

Given a font name with style appended separate into the component parts.

(e.g. "Arial Narrow Bold Italic" -> "Arial" and "Narrow Bold Italic".

**Parameters**

<i>fontfacewithstyle</i>	The name with style appended (e.g. "Arial Narrow Bold Italic")
<i>fontface</i>	The address of a <i>t_symbol</i> *. On return is set to the name of the font (e.g. "Arial")
<i>style</i>	The address of a <i>t_symbol</i> *. On return is set to the name of the font style (e.g. "Narrow Bold Italic")

### 38.57.3.18 jfont\_reference()

```
t_jfont* jfont_reference (
    t_jfont * font )
```

Create new reference to an existing font object.

#### Parameters

<i>font</i>	The font object for which to obtain a reference.
-------------	--

#### Returns

The new font object reference.

### 38.57.3.19 jfont\_set\_family()

```
void jfont_set_family (
    t_jfont * font,
    t_symbol * family )
```

Set the name of the font family (e.g.

Arial).

#### Parameters

<i>font</i>	The font object.
<i>family</i>	A <a href="#">t_symbol</a> containing the name of the desired font family.

### 38.57.3.20 jfont\_set\_font\_size()

```
void jfont_set_font_size (
    t_jfont * font,
    double size )
```

Set the size of a font object.

#### Parameters

<i>font</i>	The font object.
<i>size</i>	The new size for the font object.

### 38.57.3.21 jfont\_set\_slant()

```
void jfont_set_slant (
    t_jfont * font,
    t_jgraphics_font_slant slant )
```

Set the slant of the font.

#### Parameters

<i>font</i>	The font object
<i>slant</i>	The desired slant.

### 38.57.3.22 jfont\_set\_style()

```
void jfont_set_style (
    t_jfont * font,
    t_symbol * style )
```

Set the style of a font family (e.g.

Narrow Bold Italic).

#### Parameters

<i>font</i>	The font object.
<i>style</i>	The desired style.

### 38.57.3.23 jfont\_set\_underline()

```
void jfont_set_underline (
    t_jfont * font,
    char ul )
```

Set the underlining of a font object.

#### Parameters

<i>font</i>	The font object.
<i>ul</i>	Pass true to underline, or false for no underlining.

### 38.57.3.24 jfont\_set\_weight()

```
void jfont_set_weight (
    t_jfont * font,
    t_jgraphics_font_weight weight )
```

Set the weight of the font.

#### Parameters

<i>font</i>	The font object
<i>weight</i>	The desired weight (e.g. bold).

### 38.57.3.25 jfont\_text\_measure()

```
void jfont_text_measure (
    t_jfont * font,
    const char * utf8,
    double * width,
    double * height )
```

Given a font, find out how much area is required to render a string of text.

#### Parameters

<i>font</i>	The font object.
<i>utf8</i>	The text whose rendering will be measured.
<i>width</i>	The address of a variable to hold the width upon return.
<i>height</i>	The address of a variable to hold the height upon return.

### 38.57.3.26 jfont\_text\_measuretext\_wrapped()

```
void jfont_text_measuretext_wrapped (
    t_jfont * font,
    const char * utf8,
    double wrapwidth,
    long includewhitespace,
    double * width,
```

```
double * height,
long * numlines )
```

Given a font, find out how much area is required to render a string of text, provided a horizontal maximum limit at which the text is wrapped.

#### Parameters

<i>font</i>	The font object.
<i>utf8</i>	The text whose rendering will be measured.
<i>wrapwidth</i>	The maximum width, above which text should wrap onto a new line.
<i>includewhitespace</i>	If non-zero, include whitespace in the measurement.
<i>width</i>	The address of a variable to hold the width upon return.
<i>height</i>	The address of a variable to hold the height upon return.
<i>numlines</i>	The address of a variable to hold the number of lines of text after wrapping upon return.

### 38.57.3.27 systemfontname()

```
const char* systemfontname (
    void )
```

Retrieve the name of Max's system font.

#### Returns

The name of Max's system font.

### 38.57.3.28 systemfontname\_bold()

```
const char* systemfontname_bold (
    void )
```

Retrieve the name of Max's bold system font.

#### Returns

The name of Max's bold system font.

**38.57.3.29 systemfontname\_light()**

```
const char* systemfontname_light (
    void )
```

Retrieve the name of Max's light system font.

**Returns**

The name of Max's light system font.

**38.57.3.30 systemfontsym()**

```
t_symbol* systemfontsym (
    void )
```

Retrieve the name of Max's system font as a symbol.

**Returns**

The name of Max's system font.

## 38.58 JGraphics Matrix Transformations

The [t\\_jmatrix](#) is one way to represent a transformation.

Collaboration diagram for JGraphics Matrix Transformations:



### Data Structures

- struct [t\\_jmatrix](#)

*An affine transformation (such as scale, shear, etc).*

## Functions

- void `jgraphics_matrix_init` (`t_jmatrix` \*`x`, double `xx`, double `yx`, double `xy`, double `yy`, double `x0`, double `y0`)
 

*Set a `t_jmatrix` to an affine transformation.*
- void `jgraphics_matrix_init_identity` (`t_jmatrix` \*`x`)
 

*Modify a matrix to be an identity transform.*
- void `jgraphics_matrix_init_translate` (`t_jmatrix` \*`x`, double `tx`, double `ty`)
 

*Initialize a `t_jmatrix` to translate (offset) a point.*
- void `jgraphics_matrix_init_scale` (`t_jmatrix` \*`x`, double `sx`, double `sy`)
 

*Initialize a `t_jmatrix` to scale (offset) a point.*
- void `jgraphics_matrix_init_rotate` (`t_jmatrix` \*`x`, double `radians`)
 

*Initialize a `t_jmatrix` to rotate (offset) a point.*
- void `jgraphics_matrix_translate` (`t_jmatrix` \*`x`, double `tx`, double `ty`)
 

*Apply a translation to an existing matrix.*
- void `jgraphics_matrix_scale` (`t_jmatrix` \*`x`, double `sx`, double `sy`)
 

*Apply a scaling to an existing matrix.*
- void `jgraphics_matrix_rotate` (`t_jmatrix` \*`x`, double `radians`)
 

*Apply a rotation to an existing matrix.*
- void `jgraphics_matrix_invert` (`t_jmatrix` \*`x`)
 

*Invert an existing matrix.*
- void `jgraphics_matrix_multiply` (`t_jmatrix` \*`result`, const `t_jmatrix` \*`a`, const `t_jmatrix` \*`b`)
 

*Multiply two matrices: resulting matrix has effect of first applying `a` and then applying `b`.*
- void `jgraphics_matrix_transform_point` (const `t_jmatrix` \*`matrix`, double \*`x`, double \*`y`)
 

*Transform a point using a `t_jmatrix` transformation.*

### 38.58.1 Detailed Description

The `t_jmatrix` is one way to represent a transformation.

You can use the `t_jmatrix` in the call to `jgraphics_transform()`, `jgraphics_setmatrix()`, and `jgraphics_pattern_set_matrix` for specifying transformations.

### 38.58.2 Function Documentation

#### 38.58.2.1 `jgraphics_matrix_init()`

```
void jgraphics_matrix_init (
    t_jmatrix * x,
    double xx,
    double yx,
    double xy,
    double yy,
    double x0,
    double y0 )
```

Set a `t_jmatrix` to an affine transformation.

**Parameters**

x	
xx	
yx	
xy	
yy	
x0	
y0	

**Remarks**

given x,y the matrix specifies the following transformation:

```
xnew = xx * x + xy * y + x0;
ynew = yx * x + yy * y + y0;
```

**38.58.2.2 jgraphics\_matrix\_init\_identity()**

```
void jgraphics_matrix_init_identity (
    t_jmatrix * x )
```

Modify a matrix to be an identity transform.

**Parameters**

x	The <a href="#">t_jmatrix</a> .
---	---------------------------------

**38.58.2.3 jgraphics\_matrix\_init\_rotate()**

```
void jgraphics_matrix_init_rotate (
    t_jmatrix * x,
    double radians )
```

Initialize a [t\\_jmatrix](#) to rotate (offset) a point.

**Parameters**

x	The <a href="#">t_jmatrix</a> .
radians	The angle or rotation in radians.

### 38.58.2.4 jgraphics\_matrix\_init\_scale()

```
void jgraphics_matrix_init_scale (
    t_jmatrix * x,
    double sx,
    double sy )
```

Initialize a [t\\_jmatrix](#) to scale (offset) a point.

#### Parameters

x	The <a href="#">t_jmatrix</a> .
sx	The horizontal scale factor.
sy	The vertical scale factor.

### 38.58.2.5 jgraphics\_matrix\_init\_translate()

```
void jgraphics_matrix_init_translate (
    t_jmatrix * x,
    double tx,
    double ty )
```

Initialize a [t\\_jmatrix](#) to translate (offset) a point.

#### Parameters

x	The <a href="#">t_jmatrix</a> .
tx	The amount of x-axis translation.
ty	The amount of y-axis translation.

### 38.58.2.6 jgraphics\_matrix\_invert()

```
void jgraphics_matrix_invert (
    t_jmatrix * x )
```

Invert an existing matrix.

#### Parameters

x	The <a href="#">t_jmatrix</a> .
---	---------------------------------

### 38.58.2.7 jgraphics\_matrix\_multiply()

```
void jgraphics_matrix_multiply (
    t_jmatrix * result,
    const t_jmatrix * a,
    const t_jmatrix * b )
```

Multiply two matrices: resulting matrix has effect of first applying a and then applying b.

#### Parameters

<i>result</i>	The resulting product <a href="#">t_jmatrix</a> .
<i>a</i>	The first operand.
<i>b</i>	The second operand.

### 38.58.2.8 jgraphics\_matrix\_rotate()

```
void jgraphics_matrix_rotate (
    t_jmatrix * x,
    double radians )
```

Apply a rotation to an existing matrix.

#### Parameters

<i>x</i>	The <a href="#">t_jmatrix</a> .
<i>radians</i>	The angle or rotation in radians.

### 38.58.2.9 jgraphics\_matrix\_scale()

```
void jgraphics_matrix_scale (
    t_jmatrix * x,
    double sx,
    double sy )
```

Apply a scaling to an existing matrix.

**Parameters**

<i>x</i>	The <a href="#">t_jmatrix</a> .
<i>sx</i>	The horizontal scale factor.
<i>sy</i>	The vertical scale factor.

**38.58.2.10 jgraphics\_matrix\_transform\_point()**

```
void jgraphics_matrix_transform_point (
    const t_jmatrix * matrix,
    double * x,
    double * y )
```

Transform a point using a [t\\_jmatrix](#) transformation.

**Parameters**

<i>matrix</i>	The <a href="#">t_jmatrix</a> .
<i>x</i>	The address of the variable holding the x coordinate.
<i>y</i>	The address of the variable holding the y coordinate.

**38.58.2.11 jgraphics\_matrix\_translate()**

```
void jgraphics_matrix_translate (
    t_jmatrix * x,
    double tx,
    double ty )
```

Apply a translation to an existing matrix.

**Parameters**

<i>x</i>	The <a href="#">t_jmatrix</a> .
<i>tx</i>	The amount of x-axis translation.
<i>ty</i>	The amount of y-axis translation.

**38.59 JPattern**

A pattern is like a brush that is used to fill a path with.

Collaboration diagram for JPattern:



## Typedefs

- `typedef struct _jpattern t_jpattern`

*An instance of a jgraphics pattern.*

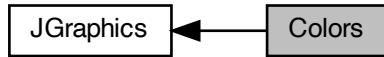
### 38.59.1 Detailed Description

A pattern is like a brush that is used to fill a path with.

It could be a solid color but it could also be an image. You can draw to a surface and then from that surface create a pattern that can be used to fill another surface. For example, `jgraphics_patter_create_for_surface()`. There are also gradients: see `jgraphics_pattern_create_linear()` and `jgraphics_pattern_create_radial()`.

## 38.60 Colors

Collaboration diagram for Colors:



## Data Structures

- `struct t_jrgb`  
*A color composed of red, green, and blue components.*
- `struct t_jrgba`  
*A color composed of red, green, blue, and alpha components.*

## Functions

- void `jrgba_to_atoms (t_jrgba *c, t_atom *argv)`  
*Get the components of a color in an array of pre-allocated atoms.*
- `t_max_err atoms_to_jrgba (long argc, t_atom *argv, t_jrgba *c)`  
*Set the components of a color by providing an array of atoms.*
- void `jrgba_set (t_jrgba *prgba, double r, double g, double b, double a)`  
*Set the components of a color.*
- void `jrgba_copy (t_jrgba *dest, t_jrgba *src)`  
*Copy a color.*
- long `jrgba_compare (t_jrgba *rgba1, t_jrgba *rgba2)`  
*Compare two colors for equality.*
- `t_max_err jrgba_attr_get (t_jrgba *jrgba, long *argc, t_atom **argv)`  
*Get the value of a `t_jrgba` struct, returned as an array of atoms with the values for each component.*
- `t_max_err jrgba_attr_set (t_jrgba *jrgba, long argc, t_atom *argv)`  
*Set the value of a `t_jrgba` struct, given an array of atoms with the values to use.*

### 38.60.1 Detailed Description

### 38.60.2 Function Documentation

#### 38.60.2.1 atoms\_to\_jrgba()

```
t_max_err atoms_to_jrgba (
    long argc,
    t_atom * argv,
    t_jrgba * c )
```

Set the components of a color by providing an array of atoms.

If it is an array of 3 atoms, then the atoms provided should define the red, green, and blue components (in this order) in a range of [0.0, 1.0]. If a 4th atom is provided, it will define the alpha channel. If the alpha channel is not defined then it is assumed to be 1.0.

#### Parameters

<code>argc</code>	The number of atoms in the array provided in <code>argv</code> . This should be 3 or 4 depending on whether or not the alpha channel is being provided.
<code>argv</code>	The address to the first of an array of atoms that define the color.
<code>c</code>	The address of a <code>t_jrgba</code> struct for which the color will be defined.

**Returns**

A Max error code.

**38.60.2.2 jrgba\_attr\_get()**

```
t_max_err jrgba_attr_get (
    t_jrgba * jrgba,
    long * argc,
    t_atom ** argv )
```

Get the value of a [t\\_jrgba](#) struct, returned as an array of atoms with the values for each component.

**Parameters**

<i>jrgba</i>	The color struct whose color will be retrieved.
<i>argc</i>	The address of a variable that will be set with the number of atoms in the argv array. The returned value should be 4. The value of the int should be set to 0 prior to calling this function.
<i>argv</i>	The address of a <a href="#">t_atom</a> pointer that will receive the a new array of atoms set to the values of the jrgba struct. The pointer should be set to NULL prior to calling this function. There should be 4 atoms returned, representing alpha, red, green, and blue components. When you are done using the atoms, you are responsible for freeing the pointer using <a href="#">sysmem_freeptr()</a> .

**Returns**

This function returns the error code [MAX\\_ERR\\_NONE](#) if successful, or one of the other error codes defined in [e\\_max\\_errorcodes](#) if unsuccessful.

**38.60.2.3 jrgba\_attr\_set()**

```
t_max_err jrgba_attr_set (
    t_jrgba * jrgba,
    long argc,
    t_atom * argv )
```

Set the value of a [t\\_jrgba](#) struct, given an array of atoms with the values to use.

**Parameters**

<i>jrgba</i>	The color struct whose color will be set.
<i>argc</i>	The number of atoms in the array. This must be 4.
<i>argv</i>	The address of the first of the atoms in the array. There must be 4 atoms, representing alpha, red, green, and blue components.

**Returns**

This function returns the error code `MAX_ERR_NONE` if successful, or one of the other error codes defined in `e_max_errorcodes` if unsuccessful.

**38.60.2.4 jrgba\_compare()**

```
long jrgba_compare (
    t_jrgba * rgba1,
    t_jrgba * rgba2 )
```

Compare two colors for equality.

**Parameters**

<code>rgba1</code>	The address of a <code>t_jrgba</code> struct to compare.
<code>rgba2</code>	The address of another <code>t_jrgba</code> struct to compare.

**Returns**

returns 1 if `rgba1 == rgba2`.

**38.60.2.5 jrgba\_copy()**

```
void jrgba_copy (
    t_jrgba * dest,
    t_jrgba * src )
```

Copy a color.

**Parameters**

<code>dest</code>	The address of a <code>t_jrgba</code> struct to which the color will be copied.
<code>src</code>	The address of a <code>t_jrgba</code> struct from which the color will be copied.

**38.60.2.6 jrgba\_set()**

```
void jrgba_set (
    t_jrgba * prgba,
```

```
double r,
double g,
double b,
double a )
```

Set the components of a color.

#### Parameters

<i>prgba</i>	The address of a <a href="#">t_jrgba</a> struct for which the color will be defined.
<i>r</i>	The value of the red component in a range of [0.0, 1.0].
<i>g</i>	The value of the green component in a range of [0.0, 1.0].
<i>b</i>	The value of the blue component in a range of [0.0, 1.0].
<i>a</i>	The value of the alpha component in a range of [0.0, 1.0].

### 38.60.2.7 [jrgba\\_to\\_atoms\(\)](#)

```
void jrgba_to_atoms (
    t_jrgba * c,
    t_atom * argv )
```

Get the components of a color in an array of pre-allocated atoms.

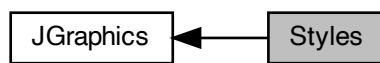
#### Parameters

<i>argv</i>	The address to the first of an array of atoms that will hold the result. At least 4 atoms must be allocated, as 4 atoms will be set by this function for the red, green, blue, and alpha components.
<i>c</i>	The address of a <a href="#">t_jrgba</a> struct from which the color components will be fetched.

## 38.61 Styles

Styles provide a means by which to inherit attribute values from a patcher that are consistently used across many objects.

Collaboration diagram for Styles:



## Macros

- `#define FILL_ATTR_SAVE`  
*Flag indicating we want this fill attribute saved (creates attrs).*
- `#define CLASS_ATTR_STYLE_RGBA_NOSAVE(c, attrname, flags, structname, structmember, label)`  
*Define an RGBA style attribute with standard settings.*
- `#define CLASS_ATTR_STYLE_RGBA(c, attrname, flags, structname, structmember, label)`  
*Define an RGBA style attribute with standard settings.*
- `#define CLASS_ATTR_STYLE_RGBA_PREVIEW(c, attrname, flags, structname, structmember, label, preview-type)`  
*Define an RGBA style attribute with standard settings.*
- `#define CLASS_ATTR_STYLE_ALIAS_NOSAVE(c, attrname, aliasname)`  
*Define an unsaved alias.*
- `#define CLASS_ATTR_STYLE_ALIAS_COMPATIBILITY(c, attrname, aliasname)`  
*Define a Max 5/6 saved compatibility alias.*
- `#define CLASS_ATTR_STYLE_ALIAS_RGBA_LEGACY(c, attrname, aliasname)`  
*Define a Max 4 legacy RGB attribute alias.*

## Functions

- `BEGIN_USING_C_LINKAGE void class_attr_setstyle (t_class *c, const char *s)`  
*Add an attribute to the current style.*
- `void class_attr_style_alias (t_class *c, const char *name, const char *aliasname, long legacy)`  
*Add an alias to the current style.*
- `void class_attr_setfill (t_class *c, const char *name, long flags)`  
*Specify that an attr requires a fill.*
- `void jgraphics_attr_fillrect (t_object *b, t_jgraphics *g, t_symbol *attrname, t_rect *area)`  
*Fill using the current value of a named style color that exists either in the object or the defined style.*
- `t_jpattern * jgraphics_attr_setfill (t_object *b, t_jgraphics *g, t_symbol *attrname, t_rect *area)`  
*Fill using the current value of a named style color that exists either in the object or the defined style.*
- `void object_attr_getfillcolor_atposition (t_object *b, const char *attrname, double pos, t_jrgba *c)`  
*Determine the color at a given position in a fill.*
- `long object_attr_getfill (t_object *obj, t_symbol *attrname)`  
*Determine if an attribute is a fill.*
- `void class_attr_stylemap (t_class *c, const char *attrname, const char *mapname)`  
*Associate the name of an attribute of your class with the name of an attribute of a style.*

### 38.61.1 Detailed Description

Styles provide a means by which to inherit attribute values from a patcher that are consistently used across many objects.

### 38.61.2 Macro Definition Documentation

### 38.61.2.1 CLASS\_ATTR\_STYLE\_ALIAS\_COMPATIBILITY

```
#define CLASS_ATTR_STYLE_ALIAS_COMPATIBILITY (
    c,
    attrname,
    aliasname )
```

Define a Max 5/6 saved compatibility alias.

#### Parameters

<i>c</i>	The class whose attribute will be added to the style.
<i>attrname</i>	The name of the attribute of your class.
<i>aliasname</i>	The name of the alias.

### 38.61.2.2 CLASS\_ATTR\_STYLE\_ALIAS\_NOSAVE

```
#define CLASS_ATTR_STYLE_ALIAS_NOSAVE (
    c,
    attrname,
    aliasname )
```

Define an unsaved alias.

#### Parameters

<i>c</i>	The class whose attribute will be added to the style.
<i>attrname</i>	The name of the attribute of your class.
<i>aliasname</i>	The name of the alias.

#### See also

'jslider' example project in the SDK.

### 38.61.2.3 CLASS\_ATTR\_STYLE\_ALIAS\_RGBA\_LEGACY

```
#define CLASS_ATTR_STYLE_ALIAS_RGBA_LEGACY (
    c,
    attrname,
    aliasname )
```

Define a Max 4 legacy RGB attribute alias.

**Parameters**

<i>c</i>	The class whose attribute will be added to the style.
<i>attrname</i>	The name of the attribute of your class.
<i>aliasname</i>	The name of the alias.

**See also**

'jslider' example project in the SDK.

**38.61.2.4 CLASS\_ATTR\_STYLE\_RGBA**

```
#define CLASS_ATTR_STYLE_RGBA(
    c,
    attrname,
    flags,
    structname,
    structmember,
    label )
```

Define an RGBA style attribute with standard settings.

**Parameters**

<i>c</i>	The class whose attribute will be added to the style.
<i>attrname</i>	The name of the attribute of your class.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>label</i>	A human-friendly label for the Max inspector.

**See also**

[CLASS\\_ATTR\\_STYLE\\_RGBA\\_NOSAVE](#) is a variant that does not save the state with the [Patcher](#).

[CLASS\\_ATTR\\_STYLE\\_RGBA\\_PREVIEW](#) is a variant that provides a style preview.

[class\\_attr\\_setstyle\(\)](#) is the lower level function used to provide the style part of the attribute definition.

**38.61.2.5 CLASS\_ATTR\_STYLE\_RGBA\_NOSAVE**

```
#define CLASS_ATTR_STYLE_RGBA_NOSAVE(
    c,
```

```
attrname,
flags,
structname,
structmember,
label )
```

Define an RGBA style attribute with standard settings.

#### Parameters

<i>c</i>	The class whose attribute will be added to the style.
<i>attrname</i>	The name of the attribute of your class.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>label</i>	A human-friendly label for the Max inspector.

#### See also

[CLASS\\_ATTR\\_STYLE\\_RGBA](#) is a variant that saves the state with the [Patcher](#).

[CLASS\\_ATTR\\_STYLE\\_RGBA\\_PREVIEW](#) is a variant that provides a style preview.

[class\\_attr\\_setstyle\(\)](#) is the lower level function used to provide the style part of the attribute definition.

### 38.61.2.6 CLASS\_ATTR\_STYLE\_RGBA\_PREVIEW

```
#define CLASS_ATTR_STYLE_RGBA_PREVIEW(
    c,
    attrname,
    flags,
    structname,
    structmember,
    label,
    previewtype )
```

Define an RGBA style attribute with standard settings.

#### Parameters

<i>c</i>	The class whose attribute will be added to the style.
<i>attrname</i>	The name of the attribute of your class.
<i>flags</i>	Any flags you wish to declare for this attribute, as defined in <a href="#">e_max_attrflags</a> .
<i>structname</i>	The C identifier for the struct (containing a valid <a href="#">t_object</a> header) representing an instance of this class.
<i>structmember</i>	The C identifier of the member in the struct that holds the value of this attribute.
<i>label</i>	A human-friendly label for the Max inspector.
<i>previewtype</i>	Type of preview to use in the style bar, e.g. "triangle_fill"

**See also**

[CLASS\\_ATTR\\_STYLE\\_RGBA\\_NOSAVE](#) is a variant that does not save the state with the [Patcher](#).

[CLASS\\_ATTR\\_STYLE\\_RGBA](#) is a variant that saves the state with the [Patcher](#) but does not provide the preview.

[class\\_attr\\_setstyle\(\)](#) is the lower level function used to provide the style part of the attribute definition.

### 38.61.2.7 FILL\_ATTR\_SAVE

```
#define FILL_ATTR_SAVE
```

Flag indicating we want this fill attribute saved (creates attrs).

**See also**

[class\\_attr\\_setfill\(\)](#)

The 'uitextfield' example project in the SDK.

## 38.61.3 Function Documentation

### 38.61.3.1 class\_attr\_setfill()

```
void class_attr_setfill (
    t_class * c,
    const char * name,
    long flags )
```

Specify that an attr requires a fill.

**Parameters**

<i>c</i>	The class whose attribute is a fill.
<i>name</i>	The name of the attribute.
<i>flags</i>	0 for none, or <a href="#">FILL_ATTR_SAVE</a> .

**See also**

The 'uitextfield' example project in the SDK.

### 38.61.3.2 class\_attr\_setstyle()

```
BEGIN_USING_C_LINKAGE void class_attr_setstyle (
    t_class * c,
    const char * s )
```

Add an attribute to the current style.

#### Parameters

<i>c</i>	The class whose attribute will be added to the style.
<i>s</i>	The name of the attribute to be added to the style.

#### See also

The 'uitextfield' example project in the SDK.

### 38.61.3.3 class\_attr\_style\_alias()

```
void class_attr_style_alias (
    t_class * c,
    const char * name,
    const char * aliasname,
    long legacy )
```

Add an alias to the current style.

This is used for backward compatibility where an attribute using an old name will want a style applied to it from a different name. Typically you will use one of the macros such as [CLASS\\_ATTR\\_STYLE\\_ALIAS\\_NOSAVE](#) rather than using this function directly.

#### Parameters

<i>c</i>	The class for whom the alias will be created.
<i>name</i>	The name of the attribute of the style.
<i>aliasname</i>	The name of the alias.
<i>legacy</i>	Always pass 0 for this argument.

#### See also

[CLASS\\_ATTR\\_STYLE\\_ALIAS\\_NOSAVE](#)

The 'jslider' project in the SDK.

### 38.61.3.4 class\_attr\_stylemap()

```
void class_attr_stylemap (
    t_class * c,
    const char * attrname,
    const char * mapname )
```

Associate the name of an attribute of your class with the name of an attribute of a style.

#### Parameters

<i>c</i>	The class whose attribute will be added to the style.
<i>attrname</i>	The name of the attribute of your class.
<i>mapname</i>	The name of the attribute from the style.

#### See also

'jslider' example project in the SDK.

### 38.61.3.5 jgraphics\_attr\_fillrect()

```
void jgraphics_attr_fillrect (
    t_object * b,
    t_jgraphics * g,
    t_symbol * attrname,
    t_rect * area )
```

Fill using the current value of a named style color that exists either in the object or the defined style.

#### Abridged example from the 'attrui' object:

```
long is_fill = object_attr_getfill(destination, x->j_attr);
if (is_fill) {
    jgraphics_attr_fillrect((t_object *)destination, g, x->j_attr, rect);
    jgraphics_rectangle(g, rect->x, rect->y, rect->width, rect->height); // ready to be stroked
}
else {
    object_attr_getjrgba(destination, x->j_attr, &color);
    jgraphics_set_source_jrgba(g, &color);
    jgraphics_rectangle(g, rect->x, rect->y, rect->width, rect->height);
    jgraphics_fill_preserve(g);
}
jgraphics_set_source_jrgba(g, &bordercolor);
jgraphics_stroke(g);
```

#### Parameters

<i>b</i>	The instance of your object.
<i>g</i>	The jgraphics context.
<i>attrname</i>	The name of the attribute whose fill style you want.
<i>area</i>	The rect area to be filled.

### 38.61.3.6 `jgraphics_attr_setfill()`

```
t_jpattern* jgraphics_attr_setfill (
    t_object * b,
    t_jgraphics * g,
    t_symbol * attrname,
    t_rect * area )
```

Fill using the current value of a named style color that exists either in the object or the defined style.

Example from the 'panel' object:

```
t_rect r;
r.x = r.y = thick * 0.5;
r.width = rect->width - thick;
r.height = rect->height - thick;
pat = jgraphics_attr_setfill((t_object *)x, g, ps_bgfillcolor, &r);
jgraphics_rectangle_rounded(g, r.x, r.y, r.width, r.height, round, round);
jgraphics_fill_preserve(g);
jgraphics_pattern_destroy(pat);
object_attr_getjrgba(x, ps_bordercolor, &color);
jgraphics_set_source_jrgba(g, &color);
jgraphics_set_line_width(g, thick);
jgraphics_stroke(g);
```

#### Parameters

<i>b</i>	The instance of your object.
<i>g</i>	The jgraphics context.
<i>attrname</i>	The name of the attribute whose fill style you want.
<i>area</i>	The rect area to be filled.

#### Returns

The pattern.

### 38.61.3.7 `object_attr_getfill()`

```
long object_attr_getfill (
    t_object * obj,
    t_symbol * attrname )
```

Determine if an attribute is a fill.

#### Parameters

<i>obj</i>	The instance of your object.
<i>attrname</i>	The name of the attribute to query.

**Returns**

true if the object is a fill, otherwise false.

**38.61.3.8 object\_attr\_getfillcolor\_atposition()**

```
void object_attr_getfillcolor_atposition (
    t_object * b,
    const char * attrname,
    double pos,
    t_jrgba * c )
```

Determine the color at a given position in a fill.

**Parameters**

<i>b</i>	The instance of your object.
<i>attrname</i>	The name of the attribute to query.
<i>pos</i>	The position in a range of [0.0, 1.0].
<i>c</i>	A valid <a href="#">t_jrgba</a> whose members will be filled-in upon return.

**Returns**

true if the object is a fill, otherwise false.

**38.62 TextField**

The textfield is a high-level text display object that may be used by a UI object to represent text in a patcher.

Collaboration diagram for TextField:



## Functions

- `t_object * textfield_get_owner (t_object *tf)`  
*Return the object that owns a particular textfield.*
- `t_max_err textfield_get_textcolor (t_object *tf, t_jrgba *prgba)`  
*Retrieve the color of the text in a textfield.*
- `t_max_err textfield_set_textcolor (t_object *tf, t_jrgba *prgba)`  
*Set the color of the text in a textfield.*
- `t_max_err textfield_get_bgcolor (t_object *tf, t_jrgba *prgba)`  
*Retrieve the background color of a textfield.*
- `t_max_err textfield_set_bgcolor (t_object *tf, t_jrgba *prgba)`  
*Set the background color of a textfield.*
- `t_max_err textfield_get_textmargins (t_object *tf, double *pleft, double *ptop, double *pright, double *pbottom)`  
*Retrieve the margins from the edge of the textfield to the text itself in a textfield.*
- `t_max_err textfield_set_textmargins (t_object *tf, double left, double top, double right, double bottom)`  
*Set the margins from the edge of the textfield to the text itself in a textfield.*
- `char textfield_get_editonclick (t_object *tf)`  
*Return the value of the 'editonclick' attribute of a textfield.*
- `t_max_err textfield_set_editonclick (t_object *tf, char c)`  
*Set the 'editonclick' attribute of a textfield.*
- `char textfield_get_selectallonedit (t_object *tf)`  
*Return the value of the 'selectallonedit' attribute of a textfield.*
- `t_max_err textfield_set_selectallonedit (t_object *tf, char c)`  
*Set the 'selectallonedit' attribute of a textfield.*
- `char textfield_get_noactivate (t_object *tf)`  
*Return the value of the 'noactivate' attribute of a textfield.*
- `t_max_err textfield_set_noactivate (t_object *tf, char c)`  
*Set the 'noactivate' attribute of a textfield.*
- `char textfield_get_READONLY (t_object *tf)`  
*Return the value of the 'readonly' attribute of a textfield.*
- `t_max_err textfield_set_READONLY (t_object *tf, char c)`  
*Set the 'readonly' attribute of a textfield.*
- `char textfield_get_wordwrap (t_object *tf)`  
*Return the value of the 'wordwrap' attribute of a textfield.*
- `t_max_err textfield_set_wordwrap (t_object *tf, char c)`  
*Set the 'wordwrap' attribute of a textfield.*
- `char textfield_get_useellipsis (t_object *tf)`  
*Return the value of the 'useellipsis' attribute of a textfield.*
- `t_max_err textfield_set_useellipsis (t_object *tf, char c)`  
*Set the 'useellipsis' attribute of a textfield.*
- `char textfield_get_autoscroll (t_object *tf)`  
*Return the value of the 'autoscroll' attribute of a textfield.*
- `t_max_err textfield_set_autoscroll (t_object *tf, char c)`  
*Set the 'autoscroll' attribute of a textfield.*
- `char textfield_get_wantsreturn (t_object *tf)`  
*Return the value of the 'wantsreturn' attribute of a textfield.*
- `t_max_err textfield_set_wantsreturn (t_object *tf, char c)`  
*Set the 'wantsreturn' attribute of a textfield.*

- `char textfield_get_wantstab (t_object *tf)`  
*Set the 'wantstab' attribute of a textfield.*
- `t_max_err textfield_set_wantstab (t_object *tf, char c)`  
*Return the value of the 'wantstab' attribute of a textfield.*
- `t_max_err textfield_set_wantstab (t_object *tf, char c)`  
*Set the 'wantstab' attribute of a textfield.*
- `char textfield_get_underline (t_object *tf)`  
*Return the value of the 'underline' attribute of a textfield.*
- `t_max_err textfield_set_underline (t_object *tf, char c)`  
*Set the 'underline' attribute of a textfield.*
- `t_max_err textfield_set_emptytext (t_object *tf, t_symbol *txt)`  
*Set the 'empty' text of a textfield.*
- `t_symbol * textfield_get_emptytext (t_object *tf)`  
*Retrieve the 'empty' text of a textfield.*

### 38.62.1 Detailed Description

The textfield is a high-level text display object that may be used by a UI object to represent text in a patcher.

It is built on the lower-level [TextLayout](#)

### 38.62.2 Function Documentation

#### 38.62.2.1 `textfield_get_autoscroll()`

```
char textfield_get_autoscroll (
    t_object * tf )
```

Return the value of the 'autoscroll' attribute of a textfield.

##### Parameters

<code>tf</code>	The textfield instance pointer.
-----------------	---------------------------------

##### Returns

A value of the attribute.

#### 38.62.2.2 `textfield_get_bgcolor()`

```
t_max_err textfield_get_bgcolor (
```

```
t_object * tf,
t_jrgba * prgba )
```

Retrieve the background color of a textfield.

#### Parameters

<i>tf</i>	The textfield instance pointer.
<i>prgba</i>	The address of a valid <a href="#">t_rgba</a> whose values will be filled-in upon return.

#### Returns

A Max error code.

### 38.62.2.3 textfield\_get\_editonclick()

```
char textfield_get_editonclick (
    t_object * tf )
```

Return the value of the 'editonclick' attribute of a textfield.

#### Parameters

<i>tf</i>	The textfield instance pointer.
-----------	---------------------------------

#### Returns

A value of the attribute.

### 38.62.2.4 textfield\_get\_emptytext()

```
t_symbol* textfield_get_emptytext (
    t_object * tf )
```

Retrieve the 'empty' text of a textfield.

The empty text is the text that is displayed in the textfield when no text is present. By default this is gensym("") .

#### Parameters

<i>tf</i>	The textfield instance pointer.
-----------	---------------------------------

**Returns**

The current text used as the empty text.

**38.62.2.5 textfield\_get\_noactivate()**

```
char textfield_get_noactivate (
    t_object * tf )
```

Return the value of the 'noactivate' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
-----------	---------------------------------

**Returns**

A value of the attribute.

**38.62.2.6 textfield\_get\_owner()**

```
t_object* textfield_get_owner (
    t_object * tf )
```

Return the object that owns a particular textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
-----------	---------------------------------

**Returns**

A pointer to the owning object.

**38.62.2.7 textfield\_get\_READONLY()**

```
char textfield_get_READONLY (
    t_object * tf )
```

Return the value of the 'readonly' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
-----------	---------------------------------

**Returns**

A value of the attribute.

**38.62.2.8 textfield\_get\_selectallonedit()**

```
char textfield_get_selectallonedit (
    t_object * tf )
```

Return the value of the 'selectallonedit' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
-----------	---------------------------------

**Returns**

A value of the attribute.

**38.62.2.9 textfield\_get\_textcolor()**

```
t_max_err textfield_get_textcolor (
    t_object * tf,
    t_jrgba * prgba )
```

Retrieve the color of the text in a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>prgba</i>	The address of a valid <a href="#">t_jrgba</a> whose values will be filled-in upon return.

**Returns**

A Max error code.

### 38.62.2.10 textfield\_get\_textmargins()

```
t_max_err textfield_get_textmargins (
    t_object * tf,
    double * pleft,
    double * ptop,
    double * pright,
    double * pbottom )
```

Retrieve the margins from the edge of the textfield to the text itself in a textfield.

#### Parameters

<i>tf</i>	The textfield instance pointer.
<i>pleft</i>	The address of a variable to hold the value of the left margin upon return.
<i>ptop</i>	The address of a variable to hold the value of the top margin upon return.
<i>pright</i>	The address of a variable to hold the value of the right margin upon return.
<i>pbottom</i>	The address of a variable to hold the value of the bottom margin upon return.

#### Returns

A Max error code.

### 38.62.2.11 textfield\_get\_underline()

```
char textfield_get_underline (
    t_object * tf )
```

Return the value of the 'underline' attribute of a textfield.

#### Parameters

<i>tf</i>	The textfield instance pointer.
-----------	---------------------------------

#### Returns

A value of the attribute.

### 38.62.2.12 textfield\_get\_useellipsis()

```
char textfield_get_useellipsis (
    t_object * tf )
```

Return the value of the 'useellipsis' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
-----------	---------------------------------

**Returns**

A value of the attribute.

**38.62.2.13 textfield\_get\_wantsreturn()**

```
char textfield_get_wantsreturn (
    t_object * tf )
```

Return the value of the 'wantsreturn' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
-----------	---------------------------------

**Returns**

A value of the attribute.

**38.62.2.14 textfield\_get\_wantstab()**

```
char textfield_get_wantstab (
    t_object * tf )
```

Return the value of the 'wantstab' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
-----------	---------------------------------

**Returns**

A value of the attribute.

### 38.62.2.15 textfield\_get\_wordwrap()

```
char textfield_get_wordwrap (
    t_object * tf )
```

Return the value of the 'wordwrap' attribute of a textfield.

#### Parameters

<i>tf</i>	The textfield instance pointer.
-----------	---------------------------------

#### Returns

A value of the attribute.

### 38.62.2.16 textfield\_set\_autoscroll()

```
t_max_err textfield_set_autoscroll (
    t_object * tf,
    char c )
```

Set the 'autoscroll' attribute of a textfield.

#### Parameters

<i>tf</i>	The textfield instance pointer.
<i>c</i>	The new value for the attribute.

#### Returns

A Max error code.

### 38.62.2.17 textfield\_set\_bgcolor()

```
t_max_err textfield_set_bgcolor (
    t_object * tf,
    t_jrgba * prgba )
```

Set the background color of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>prgba</i>	The address of a <a href="#">t_jrgba</a> containing the new color to use.

**Returns**

A Max error code.

**38.62.2.18 textfield\_set\_editonclick()**

```
t_max_err textfield_set_editonclick (
    t_object * tf,
    char c )
```

Set the 'editonclick' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>c</i>	The new value for the attribute.

**Returns**

A Max error code.

**38.62.2.19 textfield\_set\_emptytext()**

```
t_max_err textfield_set_emptytext (
    t_object * tf,
    t_symbol * txt )
```

Set the 'empty' text of a textfield.

The empty text is the text that is displayed in the textfield when no text is present. By default this is gensym("") .

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>txt</i>	A symbol containing the new text to display when the textfield has no content.

**Returns**

A Max error code.

**38.62.2.20 textfield\_set\_noactivate()**

```
t_max_err textfield_set_noactivate (
    t_object * tf,
    char c )
```

Set the 'noactivate' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>c</i>	The new value for the attribute.

**Returns**

A Max error code.

**38.62.2.21 textfield\_set\_READONLY()**

```
t_max_err textfield_set_READONLY (
    t_object * tf,
    char c )
```

Set the 'readonly' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>c</i>	The new value for the attribute.

**Returns**

A Max error code.

**38.62.2.22 textfield\_set\_selectallonedit()**

```
t_max_err textfield_set_selectallonedit (
    t_object * tf,
    char c )
```

Set the 'selectallonedit' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>c</i>	The new value for the attribute.

**Returns**

A Max error code.

**38.62.2.23 textfield\_set\_textcolor()**

```
t_max_err textfield_set_textcolor (
    t_object * tf,
    t_jrgba * prgba )
```

Set the color of the text in a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>prgba</i>	The address of a <a href="#">t_jrgba</a> containing the new color to use.

**Returns**

A Max error code.

**38.62.2.24 textfield\_set\_textmargins()**

```
t_max_err textfield_set_textmargins (
    t_object * tf,
    double left,
    double top,
    double right,
    double bottom )
```

Set the margins from the edge of the textfield to the text itself in a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>left</i>	The new value for the left margin.
<i>top</i>	The new value for the top margin.
<i>right</i>	The new value for the right margin.
<i>bottom</i>	The new value for the bottom margin.

**Returns**

A Max error code.

**38.62.2.25 textfield\_set\_underline()**

```
t_max_err textfield_set_underline (
    t_object * tf,
    char c )
```

Set the 'underline' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>c</i>	The new value for the attribute.

**Returns**

A Max error code.

**38.62.2.26 textfield\_set\_useellipsis()**

```
t_max_err textfield_set_useellipsis (
    t_object * tf,
    char c )
```

Set the 'useellipsis' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>c</i>	The new value for the attribute.

**Returns**

A Max error code.

**38.62.2.27 textfield\_set\_wantsreturn()**

```
t_max_err textfield_set_wantsreturn (
    t_object * tf,
    char c )
```

Set the 'wantsreturn' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>c</i>	The new value for the attribute.

**Returns**

A Max error code.

**38.62.2.28 textfield\_set\_wantstab()**

```
t_max_err textfield_set_wantstab (
    t_object * tf,
    char c )
```

Set the 'wantstab' attribute of a textfield.

**Parameters**

<i>tf</i>	The textfield instance pointer.
<i>c</i>	The new value for the attribute.

**Returns**

A Max error code.

### 38.62.2.29 `textfield_set_wordwrap()`

```
t_max_err textfield_set_wordwrap (
    t_object * tf,
    char c )
```

Set the 'wordwrap' attribute of a textfield.

#### Parameters

<code>tf</code>	The textfield instance pointer.
<code>c</code>	The new value for the attribute.

#### Returns

A Max error code.

## 38.63 TextLayout

A textlayout is lower-level text rendering object used by higher-level entities such as [TextField](#).

Collaboration diagram for TextLayout:



### Enumerations

- enum `t_jgraphics_textlayout_flags` { `JGRAPHICS_TEXTLAYOUT_NOWRAP` , `JGRAPHICS_TEXTLAYOUT_USEELLIPSIS` }
- Flags for setting text layout options.*

### Functions

- `t_jtextlayout * jtextlayout_create ()`  
*Create a new textlayout object.*
- `t_jtextlayout * jtextlayout_withbgcolor (t_jgraphics *g, t_jrgba *bgcolor)`  
*Create a new textlayout object.*
- `void jtextlayout_destroy (t_jtextlayout *textlayout)`

*Release/free a textlayout object.*

- void `jtextlayout_set` (`t_jtextlayout` \*textlayout, const char \*utf8, `t_jfont` \*jfont, double x, double y, double width, double height, `t_jgraphics_text_justification` justification, `t_jgraphics_textlayout_flags` flags)

*Set the text and attributes of a textlayout object.*

- void `jtextlayout_settext` (`t_jtextlayout` \*textlayout, const char \*utf8, `t_jfont` \*jfont)

*Set the text of a textlayout object.*

- void `jtextlayout_settextcolor` (`t_jtextlayout` \*textlayout, `t_jrgba` \*textcolor)

*Set the color to render text in a textlayout object.*

- void `jtextlayout_measuretext` (`t_jtextlayout` \*textlayout, long startindex, long numchars, long includewhitespace, double \*width, double \*height, long \*numlines)

*Return a measurement of how much space will be required to draw the text of a textlayout.*

- void `jtextlayout_draw` (`t_jtextlayout` \*tl, `t_jgraphics` \*g)

*Draw a textlayout in a given graphics context.*

- long `jtextlayout_getnumchars` (`t_jtextlayout` \*tl)

*Retrieve a count of the number of characters in a textlayout object.*

- `t_max_err jtextlayout_getcharbox` (`t_jtextlayout` \*tl, long index, `t_rect` \*rect)

*Retrieve the `t_rect` containing a character at a given index.*

- `t_max_err jtextlayout_getchar` (`t_jtextlayout` \*tl, long index, long \*pch)

*Retrieve the unicode character at a given index.*

- `t_jpath * jtextlayout_createpath` (`t_jtextlayout` \*tl)

*Create a `t_jpath` object representing the text layout.*

### 38.63.1 Detailed Description

A textlayout is lower-level text rendering object used by higher-level entities such as [TextField](#).

### 38.63.2 Enumeration Type Documentation

#### 38.63.2.1 `t_jgraphics_textlayout_flags`

enum `t_jgraphics_textlayout_flags`

Flags for setting text layout options.

Enumerator

<code>JGRAPHICS_TEXTLAYOUT_NOWRAP</code>	disable word wrapping
<code>JGRAPHICS_TEXTLAYOUT_USEELLIPSIS</code>	show ... if a line doesn't fit (implies NOWRAP too)

### 38.63.3 Function Documentation

#### 38.63.3.1 jtextlayout\_create()

```
t_jtextlayout* jtextlayout_create ( )
```

Create a new textlayout object.

##### Returns

The new textlayout object.

#### 38.63.3.2 jtextlayout\_createpath()

```
t_jpath* jtextlayout_createpath (
    t_jtextlayout * tl )
```

Create a t\_jpath object representing the text layout.

##### Parameters

<i>tl</i>	The textlayout object to retrieve a path for.
-----------	---

##### Returns

A t\_jpath. When finished with the path free it with jgraphics\_path\_destroy.

#### 38.63.3.3 jtextlayout\_destroy()

```
void jtextlayout_destroy (
    t_jtextlayout * textlayout )
```

Release/free a textlayout object.

##### Parameters

<i>textlayout</i>	The textlayout object to release.
-------------------	-----------------------------------

### 38.63.3.4 jtextlayout\_draw()

```
void jtextlayout_draw (
    t_jtextlayout * tl,
    t_jgraphics * g )
```

Draw a textlayout in a given graphics context.

#### Parameters

<i>tl</i>	The textlayout object to query.
<i>g</i>	The graphics context in which to draw the text.

### 38.63.3.5 jtextlayout\_getchar()

```
t_max_err jtextlayout_getchar (
    t_jtextlayout * tl,
    long index,
    long * pch )
```

Retrieve the unicode character at a given index.

#### Parameters

<i>tl</i>	The textlayout object to query.
<i>index</i>	The index from which to fetch the unicode character.
<i>pch</i>	The address of a variable to hold the unicode character value upon return.

#### Returns

A Max error code.

### 38.63.3.6 jtextlayout\_getcharbox()

```
t_max_err jtextlayout_getcharbox (
    t_jtextlayout * tl,
    long index,
    t_rect * rect )
```

Retrieve the *t\_rect* containing a character at a given index.

**Parameters**

<i>tl</i>	The textlayout object to query.
<i>index</i>	The index from which to fetch the unicode character.
<i>rect</i>	The address of a valid <code>t_rect</code> which will be filled in upon return.

**Returns**

A Max error code.

**38.63.3.7 jtextlayout\_getnumchars()**

```
long jtextlayout_getnumchars (
    t_jtextlayout * tl )
```

Retrieve a count of the number of characters in a textlayout object.

**Parameters**

<i>tl</i>	The textlayout object to query.
-----------	---------------------------------

**Returns**

The number of characters.

**38.63.3.8 jtextlayout\_measuretext()**

```
void jtextlayout_measuretext (
    t_jtextlayout * textlayout,
    long startindex,
    long numchars,
    long includewhitespace,
    double * width,
    double * height,
    long * numlines )
```

Return a measurement of how much space will be required to draw the text of a textlayout.

**Parameters**

<i>textlayout</i>	The textlayout object to query.
<i>startindex</i>	You can measure a subset of the characters. This defines the character from which to start.

**Parameters**

<i>numchars</i>	Pass -1 for all characters from startindex to end
<i>includewhitespace</i>	Define whether to measure with or without whitespace truncated from edges.
<i>width</i>	Returns the width of text not including any margins.
<i>height</i>	Returns the height of text not including any margins.
<i>numlines</i>	Returns the number of lines of text.

**38.63.3.9 jtextlayout\_set()**

```
void jtextlayout_set (
    t_jtextlayout * textlayout,
    const char * utf8,
    t_jfont * jfont,
    double x,
    double y,
    double width,
    double height,
    t_jgraphics_text_justification justification,
    t_jgraphics_textlayout_flags flags )
```

Set the text and attributes of a textlayout object.

**Parameters**

<i>textlayout</i>	The textlayout object.
<i>utf8</i>	The text to render.
<i>jfont</i>	The font with which to render the text.
<i>x</i>	The text is placed within rect specified by x, y, width, height.
<i>y</i>	The text is placed within rect specified by x, y, width, height.
<i>width</i>	The text is placed within rect specified by x, y, width, height.
<i>height</i>	The text is placed within rect specified by x, y, width, height.
<i>justification</i>	How to justify the text within the rect.
<i>flags</i>	Additional flags to control behaviour.

**38.63.3.10 jtextlayout\_settext()**

```
void jtextlayout_settext (
    t_jtextlayout * textlayout,
    const char * utf8,
    t_jfont * jfont )
```

Set the text of a textlayout object.

**Parameters**

<i>textlayout</i>	The textlayout object.
<i>utf8</i>	The text to render.
<i>jfont</i>	The font with which to render the text.

**38.63.3.11 jtextlayout\_settextcolor()**

```
void jtextlayout_settextcolor (
    t_jtextlayout * textlayout,
    t_jrgba * textcolor )
```

Set the color to render text in a textlayout object.

**Parameters**

<i>textlayout</i>	The textlayout object for which to set the color.
<i>textcolor</i>	The color for the text.

**38.63.3.12 jtextlayout\_withbgcolor()**

```
t_jtextlayout* jtextlayout_withbgcolor (
    t_jgraphics * g,
    t_jrgba * bgcolor )
```

Create a new textlayout object.

This gives a hint to the textlayout as to what the text bgcolor will be. It won't actually paint the bg for you. But, it does let it do a better job.

**Parameters**

<i>g</i>	The graphics context for the textlayout.
<i>bgcolor</i>	The background color for the textlayout.

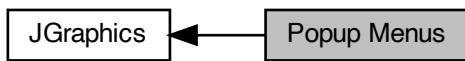
## Returns

The new textlayout object.

## 38.64 Popup Menus

Popup menu API so externals can create popup menus that can also be drawn into.

Collaboration diagram for Popup Menus:



## Functions

- `t_jpopupmenu * jpopupmenu_create ()`  
*Create a pop-up menu.*
- `void jpopupmenu_destroy (t_jpopupmenu *menu)`  
*Free a pop-up menu created with `jpopupmenu_create()`.*
- `void jpopupmenu_clear (t_jpopupmenu *menu)`  
*Clear the contents of a pop-up menu.*
- `void jpopupmenu_setcolors (t_jpopupmenu *menu, t_jrgba text, t_jrgba bg, t_jrgba highlightedtext, t_jrgba highlightedbg)`  
*Set the colors used by a pop-up menu.*
- `void jpopupmenuSetFont (t_jpopupmenu *menu, t_jfont *font)`  
*Set the font used by a pop-up menu.*
- `void jpopupmenu.AddItem (t_jpopupmenu *menu, int itemid, const char *utf8Text, t_jrgba *textColor, int checked, int disabled, t_jsurface *icon)`  
*Add an item to a pop-up menu.*
- `void jpopupmenu_Addsubmenu (t_jpopupmenu *menu, const char *utf8Name, t_jpopupmenu *submenu, int disabled)`  
*Add a pop-menu to another pop-menu as a submenu.*
- `void jpopupmenu_Addseparator (t_jpopupmenu *menu)`  
*Add a separator to a pop-menu.*
- `int jpopupmenu_popup (t_jpopupmenu *menu, t_pt screen, int defitemid)`  
*Tell a menu to display at a specified location.*
- `int jpopupmenu_popup_abovebox (t_jpopupmenu *menu, t_object *box, t_object *view, int offset, int defitemid)`  
*Tell a menu to display above a given box in a patcher.*
- `int jpopupmenu_popup_nearbox (t_jpopupmenu *menu, t_object *box, t_object *view, int defitemid)`  
*Tell a menu to display near a given box in a patcher.*

- int `jpopupmenu_popup_nearbox_with_options` (`t_jpopupmenu` \*menu, `t_object` \*box, `t_object` \*view, `t_jpopupmenu_options` opts)  
*Tell a menu to display near a given box in a patcher with options.*
- int `jpopupmenu_popup_belowrect` (`t_jpopupmenu` \*menu, `t_rect` rect, int defitemid)  
*Tell a menu to display below a given rectangle in a patcher.*
- int `jpopupmenu_popup_aboverect` (`t_jpopupmenu` \*menu, `t_rect` rect, int defitemid)  
*Tell a menu to display above a given rectangle in a patcher.*

### 38.64.1 Detailed Description

Popup menu API so externals can create popup menus that can also be drawn into.

### 38.64.2 Function Documentation

#### 38.64.2.1 `jpopupmenu_additem()`

```
void jpopupmenu_additem (
    t_jpopupmenu * menu,
    int itemid,
    const char * utf8Text,
    t_jrgba * textColor,
    int checked,
    int disabled,
    t_jsurface * icon )
```

Add an item to a pop-up menu.

##### Parameters

<code>menu</code>	The pop-up menu to which the item will be added.
<code>itemid</code>	Each menu item should be assigned a unique integer id using this parameter.
<code>utf8Text</code>	The text to display in for the menu item.
<code>textColor</code>	The color to use for the menu item, or NULL to use the default color.
<code>checked</code>	A non-zero value indicates that the item should have a check-mark next to it.
<code>disabled</code>	A non-zero value indicates that the item should be disabled.
<code>icon</code>	A <code>t_jsurface</code> will be used as an icon for the menu item if provided here. Pass NULL for no icon.

#### 38.64.2.2 `jpopupmenu_addseparator()`

```
void jpopupmenu_addseparator (
    t_jpopupmenu * menu )
```

Add a separator to a pop-menu.

#### Parameters

<i>menu</i>	The pop-up menu to which the separator will be added.
-------------	---

#### 38.64.2.3 jpopupmenu\_addsubmenu()

```
void jpopupmenu_addsubmenu (
    t_jpopupmenu * menu,
    const char * utf8Name,
    t_jpopupmenu * submenu,
    int disabled )
```

Add a pop-menu to another pop-menu as a submenu.

Note that the submenu contents are copied at the time of this call. So, any changes to the submenu after this returns won't have an effect. Also, it is safe to destroy the submenu after this function returns.

#### Parameters

<i>menu</i>	The pop-up menu to which a menu will be added as a submenu.
<i>utf8Name</i>	The name of the menu item.
<i>submenu</i>	The pop-up menu which will be used as the submenu.
<i>disabled</i>	Pass a non-zero value to disable the menu item.

#### 38.64.2.4 jpopupmenu\_clear()

```
void jpopupmenu_clear (
    t_jpopupmenu * menu )
```

Clear the contents of a pop-up menu.

#### Parameters

<i>menu</i>	The pop-up menu whose contents will be cleared.
-------------	---

#### 38.64.2.5 jpopupmenu\_create()

```
t_jpopupmenu* jpopupmenu_create ( )
```

Create a pop-up menu.

Free this pop-up menu using [jpopupmenu\\_destroy\(\)](#).

#### Returns

A pointer to the newly created jpopupmenu object.

### 38.64.2.6 [jpopupmenu\\_destroy\(\)](#)

```
void jpopupmenu_destroy (
    t_jpopupmenu * menu )
```

Free a pop-up menu created with [jpopupmenu\\_create\(\)](#).

#### Parameters

<i>menu</i>	The pop-up menu to be freed.
-------------	------------------------------

### 38.64.2.7 [jpopupmenu\\_popup\(\)](#)

```
int jpopupmenu_popup (
    t_jpopupmenu * menu,
    t_pt screen,
    int defitemid )
```

Tell a menu to display at a specified location.

#### Parameters

<i>menu</i>	The pop-up menu to display.
<i>screen</i>	The point at which to display in screen coordinates.
<i>defitemid</i>	The initially choosen item id.

#### Returns

The item id for the item in the menu choosen by the user.

### 38.64.2.8 jpopupmenu\_popup\_abovebox()

```
int jpopupmenu_popup_abovebox (
    t_jpopupmenu * menu,
    t_object * box,
    t_object * view,
    int offset,
    int defitemid )
```

Tell a menu to display above a given box in a patcher.

#### Parameters

<i>menu</i>	The pop-up menu to display.
<i>box</i>	The box above which to display the menu.
<i>view</i>	The patcherview for the box in which to display the menu.
<i>offset</i>	An offset from the box position at which to display the menu.
<i>defitemid</i>	The initially choosen item id.

#### Returns

The item id for the item in the menu choosen by the user.

### 38.64.2.9 jpopupmenu\_popup\_aboverect()

```
int jpopupmenu_popup_aboverect (
    t_jpopupmenu * menu,
    t_rect rect,
    int defitemid )
```

Tell a menu to display above a given rectangle in a patcher.

#### Parameters

<i>menu</i>	The pop-up menu to display.
<i>rect</i>	The rectangle above which to display the menu.
<i>defitemid</i>	The initially choosen item id.

#### Returns

The item id for the item in the menu choosen by the user.

### 38.64.2.10 jpopupmenu\_popup\_belowrect()

```
int jpopupmenu_popup_belowrect (
    t_jpopupmenu * menu,
    t_rect rect,
    int defitemid )
```

Tell a menu to display below a given rectangle in a patcher.

#### Parameters

<i>menu</i>	The pop-up menu to display.
<i>rect</i>	The rectangle below which to display the menu.
<i>defitemid</i>	The initially choosen item id.

#### Returns

The item id for the item in the menu choosen by the user.

### 38.64.2.11 jpopupmenu\_popup\_nearbox()

```
int jpopupmenu_popup_nearbox (
    t_jpopupmenu * menu,
    t_object * box,
    t_object * view,
    int defitemid )
```

Tell a menu to display near a given box in a patcher.

#### Parameters

<i>menu</i>	The pop-up menu to display.
<i>box</i>	The box above which to display the menu.
<i>view</i>	The patcherview for the box in which to display the menu.
<i>defitemid</i>	The initially choosen item id.

#### Returns

The item id for the item in the menu choosen by the user.

**38.64.2.12 jpopupmenu\_popup\_nearbox\_with\_options()**

```
int jpopupmenu_popup_nearbox_with_options (
    t_jpopupmenu * menu,
    t_object * box,
    t_object * view,
    t_jpopupmenu_options opts )
```

Tell a menu to display near a given box in a patcher with options.

**Parameters**

<i>menu</i>	The pop-up menu to display.
<i>box</i>	The box above which to display the menu.
<i>view</i>	The patcherview for the box in which to display the menu.
<i>opts</i>	The jpopupmenu options

**Returns**

The item id for the item in the menu chosen by the user.

**38.64.2.13 jpopupmenu\_setcolors()**

```
void jpopupmenu_setcolors (
    t_jpopupmenu * menu,
    t_jrgba text,
    t_jrgba bg,
    t_jrgba highlightedtext,
    t_jrgba highlightedbg )
```

Set the colors used by a pop-up menu.

**Parameters**

<i>menu</i>	The pop-up menu to which the colors will be applied.
<i>text</i>	The text color for menu items.
<i>bg</i>	The background color for menu items.
<i>highlightedtext</i>	The text color for the highlighted menu item.
<i>highlightedb</i>	The background color the highlighted menu item.

**38.64.2.14 jpopupmenu\_setfont()**

```
void jpopupmenu_setfont (
```

```
t_jpopupmenu * menu,
t_jfont * font )
```

Set the font used by a pop-up menu.

#### Parameters

<i>menu</i>	The pop-up menu whose font will be set.
<i>font</i>	A pointer to a font object, whose font info will be copied to the pop-up menu.

## 38.65 Box Layer

The boxlayer functions provide way to make it easier to use cached offscreen images (layers) in your drawing.

Collaboration diagram for Box Layer:



## Functions

- `t_max_err jbox_invalidate_layer (t_object *b, t_object *view, t_symbol *name)`  
*Invalidate a layer, indicating that it needs to be re-drawn.*
- `t_jgraphics * jbox_start_layer (t_object *b, t_object *view, t_symbol *name, double width, double height)`  
*Create a layer, and ready it for drawing commands.*
- `t_max_err jbox_end_layer (t_object *b, t_object *view, t_symbol *name)`  
*Conclude a layer, indicating that it is complete and ready for painting.*
- `t_max_err jbox_paint_layer (t_object *b, t_object *view, t_symbol *name, double x, double y)`  
*Paint a layer at a given position.*

### 38.65.1 Detailed Description

The boxlayer functions provide way to make it easier to use cached offscreen images (layers) in your drawing.

The general idea is to do something like this:

```
t_jgraphics *g;
g = jbox_start_layer(box, view, layername, width, height);
if (g) {
    // draw to your new offscreen context here
    // the second time you call jbox_start_layer() it will return NULL
```

```

    // since you already drew it -- you don't have to do drawing the second time
    jbox_end_layer(box, view, layername);
}
jbox_paint_layer(box, view, layername, xpos, ypos);

```

Then, if something changes where you would need to redraw the layer you invalidate it:

```
jbox_invalidate_layer(box, view, layername);
```

or

```
jbox_invalidate_layer(box, NULL, layername); // to invalidate for all views
```

Each view has its own layer stored since if a patcher has multiple views each could be at a different zoom level.

## 38.65.2 Function Documentation

### 38.65.2.1 jbox\_end\_layer()

```
t_max_err jbox_end_layer (
    t_object * b,
    t_object * view,
    t_symbol * name )
```

Conclude a layer, indicating that it is complete and ready for painting.

#### Parameters

<i>b</i>	The object/box for the layer opened by <a href="#">jbox_start_layer()</a> .
<i>view</i>	The patcherview for the object opened by <a href="#">jbox_start_layer()</a> .
<i>name</i>	The name of the layer.

#### Returns

A Max error code.

### 38.65.2.2 jbox\_invalidate\_layer()

```
t_max_err jbox_invalidate_layer (
    t_object * b,
    t_object * view,
    t_symbol * name )
```

Invalidate a layer, indicating that it needs to be re-drawn.

**Parameters**

<i>b</i>	The object/box to invalidate.
<i>view</i>	The patcherview for the object which should be invalidated, or NULL for all patcherviews.
<i>name</i>	The name of the layer to invalidate.

**Returns**

A Max error code.

**38.65.2.3 jbox\_paint\_layer()**

```
t_max_err jbox_paint_layer (
    t_object * b,
    t_object * view,
    t_symbol * name,
    double x,
    double y )
```

Paint a layer at a given position.

Note that the current color alpha value is used when painting layers to allow you to blend layers. The same is also true for [jgraphics\\_image\\_surface\\_draw\(\)](#) and [jgraphics\\_image\\_surface\\_draw\\_fast\(\)](#).

**Parameters**

<i>b</i>	The object/box to be painted.
<i>view</i>	The patcherview for the object which should be painted, or NULL for all patcherviews.
<i>name</i>	The name of the layer to paint.
<i>x</i>	The x-coordinate for the position at which to paint the layer.
<i>y</i>	The y-coordinate for the position at which to paint the layer.

**Returns**

A Max error code.

**38.65.2.4 jbox\_start\_layer()**

```
t_jgraphics* jbox_start_layer (
    t_object * b,
    t_object * view,
```

```
t_symbol * name,
double width,
double height )
```

Create a layer, and ready it for drawing commands.

The layer drawing commands must be wrapped with a matching call to [jbox\\_end\\_layer\(\)](#) prior to calling [jbox\\_paint\\_layer\(\)](#).

#### Parameters

<i>b</i>	The object/box to which the layer is attached.
<i>view</i>	The patcherview for the object to which the layer is attached.
<i>name</i>	A name for this layer.
<i>width</i>	The width of the layer.
<i>height</i>	The height of the layer.

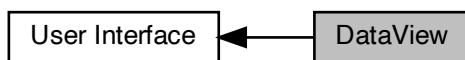
#### Returns

A [t\\_jgraphics](#) context for drawing into the layer.

## 38.66 DataView

The jdataview object provides a mechanism to display data in a tabular format.

Collaboration diagram for DataView:



## Data Structures

- struct [t\\_celldesc](#)  
*A dataview cell description.*
- struct [t\\_jcolumn](#)  
*A dataview column.*
- struct [t\\_jdataview](#)  
*The dataview object.*
- struct [t\\_privatesortrec](#)  
*used to pass data to a client sort function*

## Functions

- `void * jdataview_new (void)`  
*Create a dataview.*
- `void jdataview_setclient (t_object *dv, t_object *client)`  
*Set a dataview's client.*
- `t_object * jdataview_getclient (t_object *dv)`  
*Get a pointer to a dataview's client.*

### 38.66.1 Detailed Description

The `jdataview` object provides a mechanism to display data in a tabular format.

In Max this is used internally for the implementation of the inspectors, file browser, preferences, and `jit.cellblock` object, among others.

A `jdataview` object does not contain the information that it presents. The object you create will maintain the data and then make the data available to the dataview using the provided api.

### 38.66.2 Function Documentation

#### 38.66.2.1 `jdataview_getclient()`

```
t_object* jdataview_getclient (
    t_object * dv )
```

Get a pointer to a dataview's client.

The client is the object to which the dataview will send messages to get data, notify of changes to cells, etc.

##### Parameters

<code>dv</code>	The dataview instance.
-----------------	------------------------

##### Returns

A pointer to the dataview's client object.

#### 38.66.2.2 `jdataview_new()`

```
void* jdataview_new (
    void )
```

Create a dataview.

You should free it with [object\\_free\(\)](#).

#### Returns

A pointer to the new instance.

### 38.66.2.3 jdataview\_setclient()

```
void jdataview_setclient (
    t_object * dv,
    t_object * client )
```

Set a dataview's client.

The client is the object to which the dataview will send messages to get data, notify of changes to cells, etc. Typically this is the object in which you are creating the dataview.

#### Parameters

<i>dv</i>	The dataview instance.
<i>client</i>	The object to be assigned as the dataview's client.

## 38.67 Unicode

### Data Structures

- struct [t\\_charset\\_converter](#)  
*The charset\_converter object.*

### Functions

- [t\\_max\\_err charset\\_convert](#) ([t\\_symbol](#) \*src\_encoding, const char \*in, long inbytes, [t\\_symbol](#) \*dest\_encoding, char \*\*out, long \*outbytes)  
*A convenience function that simplifies usage by wrapping the other charset functions.*
- unsigned short \* [charset\\_utf8tounicode](#) (char \*s, long \*outlen)  
*Convert a UTF8 C-String into a 16-bit-wide-character array.*
- char \* [charset\\_unicodetoutf8](#) (unsigned short \*s, long len, long \*outlen)  
*Convert a 16-bit-wide-character array into a UTF C-string.*
- long [charset\\_utf8\\_count](#) (char \*utf8, long \*bytecount)  
*Returns utf8 character count, and optionally bytecount.*
- char \* [charset\\_utf8\\_offset](#) (char \*utf8, long charoffset, long \*byteoffset)  
*Returns utf8 character offset (positive or negative), and optionally byte offset.*

### 38.67.1 Detailed Description

### 38.67.2 Character Encodings

Currently supported character encodings

- `_sym_utf_8;` // utf-8, no bom
- `_sym_utf_16;` // utf-16, big-endian
- `_sym_utf_16be;` // utf-16, big-endian
- `_sym_utf_16le;` // utf-16, little-endian
- `_sym_iso_8859_1;` // iso-8859-1 (latin-1)
- `_sym_us_ascii;` // us-ascii 7-bit
- `_sym_ms_ansi;` // ms-ansi (microsoft code page 1252)
- `_sym_macroman;` // mac roman
- 
- `_sym_charset_converter;`
- `_sym_convert;`

#### 38.67.2.1 Example Usage

```
t_charset_converter *conv = object_new(CLASS_NOBOX, gensym("charset_converter"), ps_macroman, ps_ms_ansi);
char *cstr = "Text to convert";
char *cvtbuffer = NULL; // to-be-allocated data buffer
long cvtbuflen = 0; // length of buffer on output
if (conv) {
    // note that it isn't necessary to send in a 0-terminated string, although we do so here
    if (object_method(conv, gensym("convert")), cstr, strlen(cstr) + 1, &cvtbuffer, &cvtbuflen) == ERR_NONE) {
        // do something with the converted buffer
        system_freeptr(cvtbuffer); // free newly allocated data buffer
    }
    object_free(conv); // free converter
}
```

### 38.67.3 Function Documentation

#### 38.67.3.1 charset\_convert()

```
t_max_err charset_convert (
    t_symbol * src_encoding,
    const char * in,
    long inbytes,
    t_symbol * dest_encoding,
    char ** out,
    long * outbytes )
```

A convenience function that simplifies usage by wrapping the other charset functions.

**Parameters**

<i>src_encoding</i>	The name encoding of the input.
<i>in</i>	The input string.
<i>inbytes</i>	The number of bytes in the input string.
<i>dest_encoding</i>	The name of the encoding to use for the output.
<i>out</i>	The address of a char*, which will be allocated and filled with the string in the new encoding.
<i>outbytes</i>	The address of a value that will hold the number of bytes long the output is upon return.

**Returns**

A Max error code.

**Remarks**

Remember to call sysmem\_freeptr(\*out) to free any allocated memory.

**38.67.3.2 charset\_unicodetoutf8()**

```
char* charset_unicodetoutf8 (
    unsigned short * s,
    long len,
    long * outlen )
```

Convert a 16-bit-wide-character array into a UTF C-string.

Accepts either null termination, or not (len is zero in the latter case).

**Parameters**

<i>s</i>	An array of wide (16-bit) unicode characters.
<i>len</i>	The length of <i>s</i> .
<i>outlen</i>	The address of a variable to hold the size of the number of chars but does not include the NULL terminator in the count.

**Returns**

A UTF8-encoded C-string.

### 38.67.3.3 charset\_utf8\_count()

```
long charset_utf8_count (
    char * utf8,
    long * bytecount )
```

Returns utf8 character count, and optionally bytecount.

#### Parameters

<i>utf8</i>	The UTF-8 encoded string whose characters are to be counted.
<i>bytecount</i>	The address of a variable to hold the byte count on return. Pass NULL if you don't require the byte count.

#### Returns

The number of characters in the UTF8 string.

### 38.67.3.4 charset\_utf8\_offset()

```
char* charset_utf8_offset (
    char * utf8,
    long charoffset,
    long * byteoffset )
```

Returns utf8 character offset (positive or negative), and optionally byte offset.

#### Parameters

<i>utf8</i>	A UTF-8 encoded string.
<i>charoffset</i>	The char offset into the string at which to find the byte offset.
<i>byteoffset</i>	The address of a variable to hold the byte offset on return. Pass NULL if you don't require the byte offset.

#### Returns

The character offset.

### 38.67.3.5 charset\_utf8tounicode()

```
unsigned short* charset_utf8tounicode (
    char * s,
    long * outlen )
```

Convert a UTF8 C-String into a 16-bit-wide-character array.

**Parameters**

<i>s</i>	The string to be converted to unicode.
<i>outlen</i>	The address of a variable to hold the size of the number of chars but does not include the NULL terminator in the count.

**Returns**

A pointer to the buffer of unicode (wide) characters.

## 38.68 Atom Module

Collaboration diagram for Atom Module:



## Functions

- [t\\_jit\\_err jit\\_atom\\_setlong \(t\\_atom \\*a, t\\_atom\\_long b\)](#)  
*Sets atom value to long integer.*
- [t\\_jit\\_err jit\\_atom\\_setfloat \(t\\_atom \\*a, double b\)](#)  
*Sets atom value to floating point number.*
- [t\\_jit\\_err jit\\_atom\\_setsym \(t\\_atom \\*a, t\\_symbol \\*b\)](#)  
*Sets atom value to symbol.*
- [t\\_jit\\_err jit\\_atom\\_setobj \(t\\_atom \\*a, void \\*b\)](#)  
*Sets atom value to object pointer.*
- [t\\_atom\\_long jit\\_atom\\_getlong \(t\\_atom \\*a\)](#)  
*Retrieves atom value as long integer.*
- [double jit\\_atom\\_getfloat \(t\\_atom \\*a\)](#)  
*Retrieves atom value as floating point number.*
- [t\\_symbol \\* jit\\_atom\\_getsym \(t\\_atom \\*a\)](#)  
*Retrieves atom value as symbol pointer.*
- [void \\* jit\\_atom\\_getobj \(t\\_atom \\*a\)](#)  
*Retrieves atom value as object pointer.*
- [long jit\\_atom\\_getcharfix \(t\\_atom \\*a\)](#)  
*Retrieves atom value as an 8 bit fixed point number.*
- [long jit\\_atom\\_arg\\_getlong \(t\\_atom\\_long \\*c, long idx, long ac, t\\_atom \\*av\)](#)  
*Retrieves atom argument at index as long integer if present.*

- long `jit_atom_arg_getfloat` (float \*c, long idx, long ac, `t_atom` \*av)  
*Retrieves atom argument at index as floating point number if present.*
- long `jit_atom_arg_getdouble` (double \*c, long idx, long ac, `t_atom` \*av)  
*Retrieves atom argument at index as double precision floating point number if present.*
- long `jit_atom_arg_getsym` (`t_symbol` \*\*c, long idx, long ac, `t_atom` \*av)  
*Retrieves atom argument at index as symbol pointer if present.*

### 38.68.1 Detailed Description

### 38.68.2 Function Documentation

#### 38.68.2.1 `jit_atom_arg_getdouble()`

```
long jit_atom_arg_getdouble (
    double * c,
    long idx,
    long ac,
    t_atom * av )
```

Retrieves atom argument at index as double precision floating point number if present.

This function is useful for setting the values only if there is an argument at the specified index, otherwise, the input value is untouched.

#### Parameters

<code>c</code>	pointer to double (should contain desired default)
<code>idx</code>	atom argument index
<code>ac</code>	atom argument count
<code>av</code>	atom argument vector

**Returns**

`t_jit_err` error code. `JIT_ERR_NONE` if successful.

References `atom_arg_getdouble()`.

Here is the call graph for this function:



### 38.68.2.2 `jit_atom_arg_getfloat()`

```
long jit_atom_arg_getfloat (
    float * c,
    long idx,
    long ac,
    t_atom * av )
```

Retrieves atom argument at index as floating point number if present.

This function is useful for setting the values only if there is an argument at the specified index, otherwise, the input value is untouched.

**Parameters**

<code>c</code>	pointer to float (should contain desired default)
<code>idx</code>	atom argument index
<code>ac</code>	atom argument count
<code>av</code>	atom argument vector

**Returns**

`t_jit_err` error code. `JIT_ERR_NONE` if successful.

References `atom_arg_getfloat()`.

Here is the call graph for this function:



### 38.68.2.3 jit\_atom\_arg\_getlong()

```
long jit_atom_arg_getlong (
    t_atom_long * c,
    long idx,
    long ac,
    t_atom * av )
```

Retrieves atom argument at index as long integer if present.

This function is useful for setting the values only if there is an argument at the specified index, otherwise, the input value is untouched.

#### Parameters

<i>c</i>	pointer to long (should contain desired default)
<i>idx</i>	atom argument index
<i>ac</i>	atom argument count
<i>av</i>	atom argument vector

#### Returns

t\_jit\_err error code. JIT\_ERR\_NONE if successful.

References atom\_arg\_getlong().

Referenced by max\_jit\_mop\_matrix\_args().

Here is the call graph for this function:



#### 38.68.2.4 jit\_atom\_arg\_getsym()

```
long jit_atom_arg_getsym (
    t_symbol ** c,
    long idx,
    long ac,
    t_atom * av )
```

Retrieves atom argument at index as symbol pointer if present.

This function is useful for setting the values only if there is an argument at the specified index, otherwise, the input value is untouched.

##### Parameters

<i>c</i>	pointer to symbol pointer (should contain desired default)
<i>idx</i>	atom argument index
<i>ac</i>	atom argument count
<i>av</i>	atom argument vector

##### Returns

*t\_jit\_err* error code. *JIT\_ERR\_NONE* if successful.

References *atom\_arg\_getsym()*.

Referenced by *max\_jit\_mop\_matrix\_args()*.

Here is the call graph for this function:



### 38.68.2.5 jit\_atom\_getcharfix()

```
long jit_atom_getcharfix (
    t_atom * a )
```

Retrieves atom value as an 8 bit fixed point number.

#### Parameters

a	atom pointer
---	--------------

#### Returns

8 bit fixed point value in the range 0-255. 0 if atom has no numeric value.

References atom\_getcharfix().

Referenced by jit\_matrix\_fillplane(), jit\_matrix\_setall(), and jit\_matrix\_setcell().

Here is the call graph for this function:



### 38.68.2.6 jit\_atom\_getfloat()

```
double jit_atom_getfloat (
    t_atom * a )
```

Retrieves atom value as floating point number.

#### Parameters

a	atom pointer
---	--------------

#### Returns

floating point value. 0 if atom has no numeric value.

References atom\_getfloat().

Referenced by jit\_matrix\_fillplane(), jit\_matrix\_setall(), and jit\_matrix\_setcell().

Here is the call graph for this function:



### 38.68.2.7 jit\_atom\_getlong()

```
long jit_atom_getlong (
    t_atom * a )
```

Retrieves atom value as long integer.

#### Parameters

a	atom pointer
---	--------------

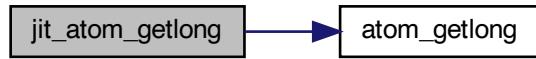
**Returns**

long integer value. 0 if atom has no numeric value.

References atom\_getlong().

Referenced by jit\_matrix\_exprfill(), jit\_matrix\_fillplane(), jit\_matrix\_getcell(), jit\_matrix\_setall(), jit\_matrix\_setcell(), jit\_matrix\_setcell1d(), jit\_matrix\_setcell2d(), jit\_matrix\_setcell3d(), jit\_matrix\_setplane1d(), jit\_matrix\_setplane2d(), and jit\_matrix\_setplane3d().

Here is the call graph for this function:



### 38.68.2.8 jit\_atom\_getobj()

```
void* jit_atom_getobj (
    t_atom * a )
```

Retrieves atom value as object pointer.

**Parameters**

a	atom pointer
---	--------------

**Returns**

object pointer. NULL if atom has no object value.

References atom\_getobj().

Referenced by jit\_matrix\_setcell().

Here is the call graph for this function:



### 38.68.2.9 jit\_atom\_getsym()

```
t_symbol * jit_atom_getsym ( t_atom * a )
```

Retrieves atom value as symbol pointer.

#### Parameters

a	atom pointer
---	--------------

#### Returns

symbol pointer. `_jit_sym_nothing` if atom has no symbolic value.

References `_jit_sym_nothing`, and `atom_getsym()`.

Referenced by `jit_matrix_exprfill()`, `jit_matrix_jit_gl_texture()`, `jit_matrix_op()`, `jit_matrix_setcell()`, and `max_jit_mop_jit_matrix()`.

Here is the call graph for this function:



### 38.68.2.10 jit\_atom\_setfloat()

```
t_jit_err jit_atom_setfloat (
    t_atom * a,
    double b )
```

Sets atom value to floating point number.

#### Parameters

<i>a</i>	atom pointer
<i>b</i>	floating point value

#### Returns

*t\_jit\_err* error code.

References `atom_setfloat()`.

Referenced by `jit_matrix_getcell()`.

Here is the call graph for this function:



### 38.68.2.11 jit\_atom\_setlong()

```
t_jit_err jit_atom_setlong (
    t_atom * a,
    t_atom_long b )
```

Sets atom value to long integer.

#### Parameters

<i>a</i>	atom pointer
<i>b</i>	integer value

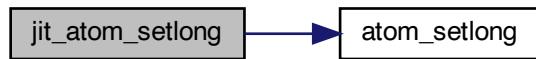
**Returns**

t\_jit\_err error code.

References atom\_setlong().

Referenced by jit\_matrix\_getcell(), jit\_matrix\_setcell1d(), jit\_matrix\_setcell2d(), jit\_matrix\_setcell3d(), jit\_matrix\_setplane1d(), jit\_matrix\_setplane2d(), and jit\_matrix\_setplane3d().

Here is the call graph for this function:



### 38.68.2.12 jit\_atom\_setobj()

```
t_jit_err jit_atom_setobj (
    t_atom * a,
    void * b )
```

Sets atom value to object pointer.

**Parameters**

a	atom pointer
b	object pointer

**Returns**

t\_jit\_err error code.

References atom\_setobj().

Referenced by jit\_matrix\_getcell(), and jit\_matrix\_jit\_gl\_texture().

Here is the call graph for this function:



### 38.68.2.13 jit\_atom\_setsym()

```
t_jit_err jit_atom_setsym (
    t_atom * a,
    t_symbol * b )
```

Sets atom value to symbol.

#### Parameters

<i>a</i>	atom pointer
<i>b</i>	symbol value

#### Returns

*t\_jit\_err* error code.

References `atom_setsym()`.

Referenced by `jit_matrix_getcell()`, `jit_matrix_setcell1d()`, `jit_matrix_setcell2d()`, `jit_matrix_setcell3d()`, `jit_matrix_setplane1d()`, `jit_matrix_setplane2d()`, `jit_matrix_setplane3d()`, `jit_mop_single_type()`, `jit_object_exportattrs()`, and `max_jit_mop_outputmatrix()`.

Here is the call graph for this function:



## 38.69 Attribute Module

Collaboration diagram for Attribute Module:



### Data Structures

- struct `t_jit_attribute`  
`t_jit_attribute` object struct.
- struct `t_jit_attr_offset`  
`t_jit_attr_offset` object struct.
- struct `t_jit_attr_offset_array`  
`t_jit_attr_offset_array` object struct.
- struct `t_jit_attr_filter_clip`  
`t_jit_attr_filter_clip` object struct.
- struct `t_jit_attr_filter_proc`  
`t_jit_attr_filter_proc` object struct.
- struct `t_jit_attr`  
`Common attribute struct.`

### Functions

- `t_symbol * jit_attr_getname (t_jit_attr *x)`  
*Retrieves attribute name.*
- `t_symbol * jit_attr_gettype (t_jit_attr *x)`  
*Retrieves attribute type.*
- `t_atom_long jit_attr_caget (t_jit_attr *x)`  
*Retrieves attribute gettable flag.*
- `t_atom_long jit_attr_canset (t_jit_attr *x)`  
*Retrieves attribute settable flag.*
- `t_atom_long jit_attr_usercaget (t_jit_attr *x)`  
*Retrieves attribute user gettable flag.*
- `t_atom_long jit_attr_usercanset (t_jit_attr *x)`  
*Retrieves attribute user settable flag.*
- `method jit_attr_getmethod (t_jit_attr *x, t_symbol *methodname)`  
*Retrieves attribute getter or setter method.*
- `t_jit_err jit_attr_filterget (t_jit_attr *x, void *y)`

- `t_jit_err jit_attr_filterset (t_jit_attr *x, void *y)`

*Sets attribute getter filter.*
- `t_jit_err jit_attr_get (t_jit_attr *x, void *parent, long *ac, t_atom **av)`

*Sets attribute setter filter.*
- `t_jit_err jit_attr_set (t_jit_attr *x, void *parent, long ac, t_atom *av)`

*Calls attribute getter to retrieve from parent object.*
- `t_jit_object * jit_attribute_new (char *name, t_symbol *type, long flags, method mget, method mset)`

*Constructs instance of `t_jit_attribute`.*
- `t_jit_object * jit_attr_offset_new (char *name, t_symbol *type, long flags, method mget, method mset, long offset)`

*Constructs instance of `t_jit_attr_offset`.*
- `t_jit_object * jit_attr_offset_array_new (char *name, t_symbol *type, long size, long flags, method mget, method mset, long offsetcount, long offset)`

*Constructs instance of `t_jit_attr_offset_array`.*
- `t_jit_object * jit_attr_filter_clip_new (void)`

*Constructs instance of `t_jit_attr_filter_clip`.*
- `t_jit_object * jit_attr_filter_proc_new (method proc)`

*Constructs instance of `t_jit_attr_filter_proc`.*
- `t_atom_long jit_attr_getlong (void *x, t_symbol *s)`

*Retrieves attribute value as a long integer value.*
- `t_jit_err jit_attr_setlong (void *x, t_symbol *s, t_atom_long c)`

*Sets attribute value as a long integer value.*
- `t_atom_float jit_attr_getfloat (void *x, t_symbol *s)`

*Retrieves attribute value as a floating point value.*
- `t_jit_err jit_attr_setfloat (void *x, t_symbol *s, t_atom_float c)`

*Sets attribute value as a floating point value.*
- `t_symbol * jit_attr_getsym (void *x, t_symbol *s)`

*Retrieves attribute value as a symbol value.*
- `t_jit_err jit_attr_setsym (void *x, t_symbol *s, t_symbol *c)`

*Sets attribute value as a symbol value.*
- `long jit_attr_getlong_array (void **x, t_symbol *s, long max, t_atom_long *vals)`

*Retrieves attribute value as an array of long integer values.*
- `t_jit_err jit_attr_setlong_array (void **x, t_symbol *s, long count, t_atom_long *vals)`

*Sets attribute value as an array of long integer values.*
- `long jit_attr_getchar_array (void **x, t_symbol *s, long max, uchar *vals)`

*Retrieves attribute value as an array of char values.*
- `t_jit_err jit_attr_setchar_array (void **x, t_symbol *s, long count, uchar *vals)`

*Sets attribute value as an array of char values.*
- `long jit_attr_getfloat_array (void **x, t_symbol *s, long max, float *vals)`

*Retrieves attribute value as an array of floating point values.*
- `t_jit_err jit_attr_setfloat_array (void **x, t_symbol *s, long count, float *vals)`

*Sets attribute value as an array of floating point values.*
- `long jit_attr_getdouble_array (void **x, t_symbol *s, long max, double *vals)`

*Retrieves attribute value as an array of double precision floating point values.*
- `t_jit_err jit_attr_setdouble_array (void **x, t_symbol *s, long count, double *vals)`

*Sets attribute value as an array of double precision floating point values.*
- `long jit_attr_getsym_array (void **x, t_symbol *s, long max, t_symbol **vals)`

*Retrieves attribute value as an array of symbol values.*

- `t_jit_err jit_attr_setsym_array (void *x, t_symbol *s, long count, t_symbol **vals)`  
*Sets attribute value as an array of symbol values.*
- `long jit_attr_symcompare (void *x, t_symbol *name)`  
*Compares symbol name with name provided.*

### 38.69.1 Detailed Description

### 38.69.2 Function Documentation

#### 38.69.2.1 jit\_attr\_canget()

```
t_atom_long jit_attr_canget (
    t_jit_attr * x )
```

Retrieves attribute gettable flag.

##### Parameters

<code>x</code>	attribute object pointer
----------------	--------------------------

##### Returns

gettable flag

##### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of any attribute object.

References `t_jit_attr::flags`, and `JIT_ATTR_GET_OPAQUE`.

#### 38.69.2.2 jit\_attr\_canset()

```
t_atom_long jit_attr_canset (
    t_jit_attr * x )
```

Retrieves attribute settable flag.

**Parameters**

x	attribute object pointer
---	--------------------------

**Returns**

settable flag

**Warning**

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of any attribute obejct.

References `t_jit_attr::flags`, and `JIT_ATTR_SET_OPAQUE`.

### 38.69.2.3 `jit_attr_filter_clip_new()`

```
t_jit_object * jit_attr_filter_clip_new (
    void )
```

Constructs instance of [t\\_jit\\_attr\\_filter\\_clip](#).

**Returns**

`t_jit_attr_filter_clip` object pointer

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_new`.

References `_jit_sym_float64`, `t_jit_attr_filter_clip::max`, `t_jit_attr_filter_clip::min`, `t_jit_attr_filter_clip::scale`, `t_jit_attr_filter_clip::type`, `t_jit_attr_filter_clip::usemax`, `t_jit_attr_filter_clip::usemin`, and `t_jit_attr_filter_clip::usescale`.

### 38.69.2.4 `jit_attr_filter_proc_new()`

```
t_jit_object * jit_attr_filter_proc_new (
    method proc )
```

Constructs instance of [t\\_jit\\_attr\\_filter\\_proc](#).

**Parameters**

<i>proc</i>	filter procedure
-------------	------------------

**Returns**

[t\\_jit\\_attr\\_filter\\_clip](#) object pointer

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_new`.

References `t_jit_attr_filter_proc::proc`.

**38.69.2.5 jit\_attr\_filterget()**

```
t_jit_err jit_attr_filterget (
    t_jit_attr * x,
    void * y )
```

Sets attribute getter filter.

**Parameters**

<i>x</i>	attribute object pointer
<i>y</i>	getter filter object

**Returns**

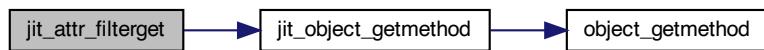
`t_jit_err` error code

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of any attribute object.

References `_jit_sym_filter`, `t_jit_attr::filterget`, and `jit_object_getmethod()`.

Here is the call graph for this function:



### 38.69.2.6 jit\_attr\_filterset()

```
t_jit_err jit_attr_filterset (
    t_jit_attr * x,
    void * y )
```

Sets attribute setter filter.

#### Parameters

x	attribute object pointer
y	setter filter object

#### Returns

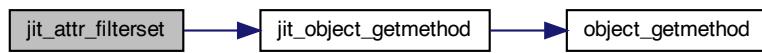
t\_jit\_err error code

#### Warning

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of any attribute obejct.

References \_jit\_sym\_filter, t\_jit\_attr::filterset, and jit\_object\_getmethod().

Here is the call graph for this function:



### 38.69.2.7 jit\_attr\_get()

```
t_jit_err jit_attr_get (
    t_jit_attr * x,
    void * parent,
    long * ac,
    t_atom ** av )
```

Calls attribute getter to retrieve from parent object.

**Parameters**

<i>x</i>	attribute object pointer
<i>parent</i>	target object pointer
<i>ac</i>	pointer to argument count
<i>av</i>	pointer to argument vector

**Returns**

`t_jit_err` error code

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of any attribute object.

References `t_jit_attr::filterget`, and `t_jit_attr::get`.

**38.69.2.8 `jit_attr_getchar_array()`**

```
long jit_attr_getchar_array (
    void * x,
    t_symbol * s,
    long max,
    uchar * vals )
```

Retrieves attribute value as an array of char values.

**Parameters**

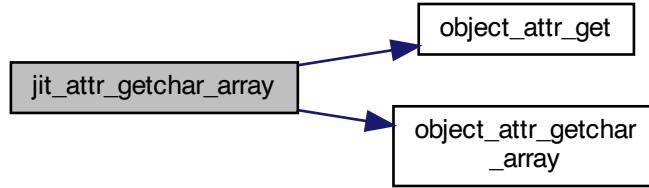
<i>x</i>	object pointer
<i>s</i>	attribute name
<i>max</i>	maximum number of values to copy
<i>vals</i>	pointer to retrieved values

**Returns**

number of values retrieved.

References `object_attr_get()`, and `object_attr_getchar_array()`.

Here is the call graph for this function:



### 38.69.2.9 jit\_attr\_getdouble\_array()

```
long jit_attr_getdouble_array (
    void * x,
    t_symbol * s,
    long max,
    double * vals )
```

Retrieves attribute value as an array of double precision floating point values.

#### Parameters

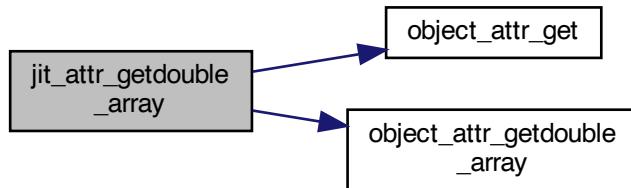
<i>x</i>	object pointer
<i>s</i>	attribute name
<i>max</i>	maximum number of values to copy
<i>vals</i>	pointer to retrieved values

**Returns**

number of values retrieved.

References `object_attr_get()`, and `object_attr_getdouble_array()`.

Here is the call graph for this function:



### 38.69.2.10 jit\_attr\_getfloat()

```
t_atom_float jit_attr_getfloat (
    void * x,
    t_symbol * s )
```

Retrieves attribute value as a floating point value.

**Parameters**

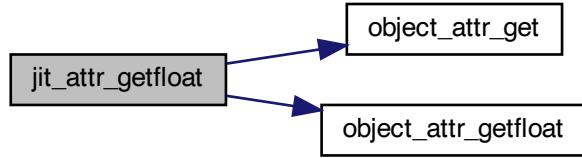
x	object pointer
s	attribute name

**Returns**

floating point value

References `object_attr_get()`, and `object_attr_getfloat()`.

Here is the call graph for this function:



### 38.69.2.11 jit\_attr\_getfloat\_array()

```
long jit_attr_getfloat_array (
    void * x,
    t_symbol * s,
    long max,
    float * vals )
```

Retrieves attribute value as an array of floating point values.

#### Parameters

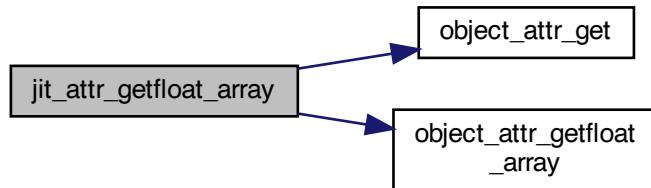
<i>x</i>	object pointer
<i>s</i>	attribute name
<i>max</i>	maximum number of values to copy
<i>vals</i>	pointer to retrieved values

**Returns**

number of values retrieved.

References `object_attr_get()`, and `object_attr_getfloat_array()`.

Here is the call graph for this function:

**38.69.2.12 jit\_attr\_getlong()**

```
t_atom_long jit_attr_getlong (
    void * x,
    t_symbol * s )
```

Retrieves attribute value as a long integer value.

**Parameters**

<code>x</code>	object pointer
<code>s</code>	attribute name

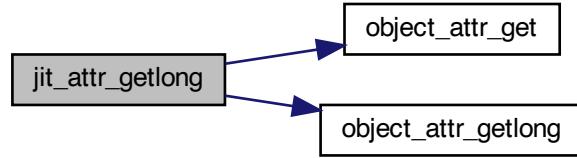
**Returns**

long integer value

References `object_attr_get()`, and `object_attr_getlong()`.

Referenced by `max_jit_classex_mop_wrap()`, `max_jit_mop_adapt_matrix_all()`, `max_jit_mop_clear()`, `max_jit_mop_free()`, `max_jit_mop_getoutputmode()`, `max_jit_mop_inputs()`, `max_jit_mop_jit_matrix()`, `max_jit_mop_matrix_args()`, `max_jit_mop_notify()`, `max_jit_mop_outputmatrix()`, and `max_jit_mop_outputs()`.

Here is the call graph for this function:



### 38.69.2.13 jit\_attr\_getlong\_array()

```
long jit_attr_getlong_array (
    void * x,
    t_symbol * s,
    long max,
    t_atom_long * vals )
```

Retrieves attribute value as an array of long integer values.

#### Parameters

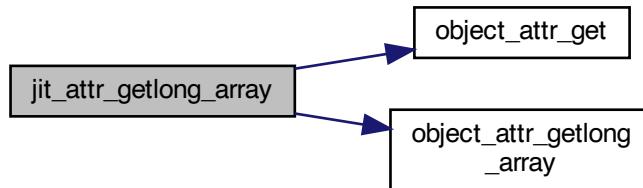
<i>x</i>	object pointer
<i>s</i>	attribute name
<i>max</i>	maximum number of values to copy
<i>vals</i>	pointer to retrieved values

**Returns**

number of values retrieved.

References `object_attr_get()`, and `object_attr_getlong_array()`.

Here is the call graph for this function:

**38.69.2.14 jit\_attr\_getmethod()**

```
method jit_attr_getmethod (
    t_jit_attr * x,
    t_symbol * methodname )
```

Retrieves attribute getter or setter method.

**Parameters**

<i>x</i>	attribute object pointer
<i>methodname</i>	"get" or "set" symbol

**Returns**

getter or setter method

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of any attribute object.

References `_jit_sym_get`, `_jit_sym_set`, `t_jit_attr::filterget`, `t_jit_attr::filterset`, `t_jit_attr::get`, and `t_jit_attr::set`.

### 38.69.2.15 jit\_attr\_getname()

```
t_symbol * jit_attr_getname (
    t_jit_attr * x )
```

Retrieves attribute name.

#### Parameters

x	attribute object pointer
---	--------------------------

#### Returns

attribute name

#### Warning

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of any attribute object.

References \_jit\_sym\_nothing, and t\_jit\_attr::name.

### 38.69.2.16 jit\_attr\_getsym()

```
t_symbol* jit_attr_getsym (
    void * x,
    t_symbol * s )
```

Retrieves attribute value as a symbol value.

#### Parameters

x	object pointer
s	attribute name

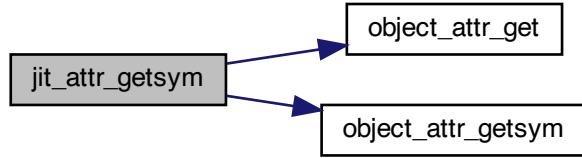
#### Returns

symbol value

References object\_attr\_get(), and object\_attr\_getsym().

Referenced by jit\_mop\_io\_restrict\_type(), max\_jit\_classex\_mop\_wrap(), max\_jit\_mop\_assist(), max\_jit\_mop\_free(), max\_jit\_mop\_notify(), and max\_jit\_mop\_outputmatrix().

Here is the call graph for this function:



### 38.69.2.17 jit\_attr\_getsym\_array()

```
long jit_attr_getsym_array (
    void * x,
    t_symbol * s,
    long max,
    t_symbol ** vals )
```

Retrieves attribute value as an array of symbol values.

#### Parameters

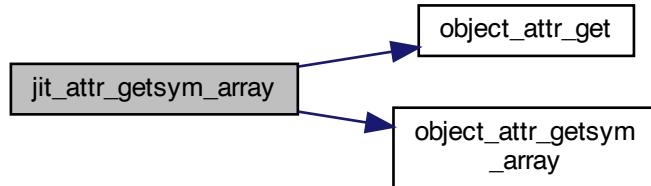
<i>x</i>	object pointer
<i>s</i>	attribute name
<i>max</i>	maximum number of values to copy
<i>vals</i>	pointer to retrieved values

**Returns**

number of values retrieved.

References `object_attr_get()`, and `object_attr_getsym_array()`.

Here is the call graph for this function:

**38.69.2.18 jit\_attr\_gettype()**

```
t_symbol * jit_attr_gettype (
    t_jit_attr * x )
```

Retrieves attribute type.

**Parameters**

<code>x</code>	attribute object pointer
----------------	--------------------------

**Returns**

attribute type

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of any attribute object.

References `_jit_sym_nothing`, and `t_jit_attr::type`.

### 38.69.2.19 jit\_attr\_offset\_array\_new()

```
t_jit_object * jit_attr_offset_array_new (
    char * name,
    t_symbol * type,
    long size,
    long flags,
    method mget,
    method mset,
    long offsetcount,
    long offset )
```

Constructs instance of [t\\_jit\\_attr\\_offset\\_array](#).

#### Parameters

<i>name</i>	attribute name
<i>type</i>	data type
<i>size</i>	maximum size
<i>flags</i>	privacy flags
<i>mget</i>	getter method
<i>mset</i>	setter method
<i>offsetcount</i>	byte offset to count struct member (if zero, remain fixed size with max size)
<i>offset</i>	byte offset to array struct member

#### Returns

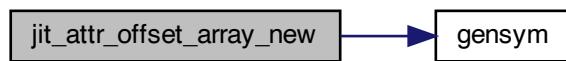
[t\\_jit\\_attr\\_offset\\_array](#) object pointer

#### Warning

This function is not exported, but is provided for reference when calling via `jit_object_new`.

References `t_jit_attr_offset_array::filterget`, `t_jit_attr_offset_array::filterset`, `t_jit_attr_offset_array::flags`, `gensym()`, `t_jit_attr_offset_array::get`, `t_jit_attr_offset_array::name`, `t_jit_attr_offset_array::offset`, `t_jit_attr_offset_array::offsetcount`, `t_jit_attr_offset_array::reserved`, `t_jit_attr_offset_array::set`, `t_jit_attr_offset_array::size`, and `t_jit_attr_offset_array::type`.

Here is the call graph for this function:



### 38.69.2.20 jit\_attr\_offset\_new()

```
t_jit_object * jit_attr_offset_new (
    char * name,
    t_symbol * type,
    long flags,
    method mget,
    method mset,
    long offset )
```

Constructs instance of [t\\_jit\\_attr\\_offset](#).

#### Parameters

<i>name</i>	attribute name
<i>type</i>	data type
<i>flags</i>	privacy flags
<i>mget</i>	getter method
<i>mset</i>	setter method
<i>offset</i>	byte offset to struct member

#### Returns

[t\\_jit\\_attr\\_offset](#) object pointer

#### Warning

This function is not exported, but is provided for reference when calling via `jit_object_new`.

References `t_jit_attr_offset::filterget`, `t_jit_attr_offset::filterset`, `t_jit_attr_offset::flags`, `gensym()`, `t_jit_attr_offset::get`, `t_jit_attr_offset::name`, `t_jit_attr_offset::offset`, `t_jit_attr_offset::reserved`, `t_jit_attr_offset::set`, and `t_jit_attr_offset::type`.

Here is the call graph for this function:



### 38.69.2.21 jit\_attr\_set()

```
t_jit_err jit_attr_set (
    t_jit_attr * x,
    void * parent,
    long ac,
    t_atom * av )
```

Calls attribute setter to set in parent object.

#### Parameters

<i>x</i>	attribute object pointer
<i>parent</i>	target object pointer
<i>ac</i>	argument count
<i>av</i>	argument vector

#### Returns

`t_jit_err` error code

#### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of any attribute object.

References `t_jit_attr::filterset`, and `t_jit_attr::set`.

### 38.69.2.22 jit\_attr\_setchar\_array()

```
t_jit_err jit_attr_setchar_array (
    void * x,
    t_symbol * s,
    long count,
    uchar * vals )
```

Sets attribute value as an array of char values.

#### Parameters

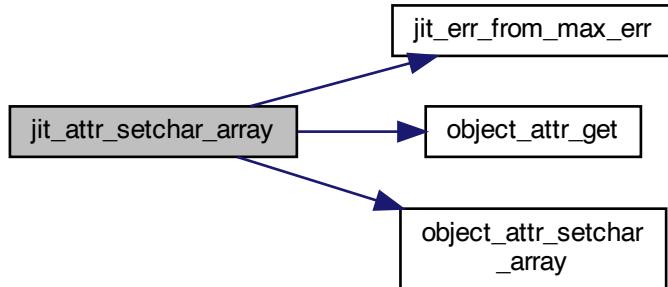
<i>x</i>	object pointer
<i>s</i>	attribute name
<i>count</i>	number of values
<i>vals</i>	pointer to values

**Returns**

`t_jit_err` error code.

References `jit_err_from_max_err()`, `object_attr_get()`, and `object_attr_setchar_array()`.

Here is the call graph for this function:

**38.69.2.23 `jit_attr_setdouble_array()`**

```
t_jit_err jit_attr_setdouble_array (
    void * x,
    t_symbol * s,
    long count,
    double * vals )
```

Sets attribute value as an array of double precision floating point values.

**Parameters**

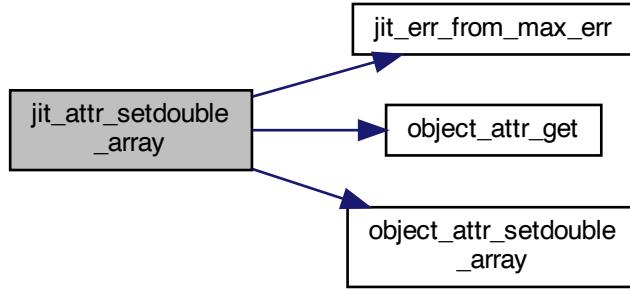
<code>x</code>	object pointer
<code>s</code>	attribute name
<code>count</code>	number of values
<code>vals</code>	pointer to values

**Returns**

`t_jit_err` error code.

References `jit_err_from_max_err()`, `object_attr_get()`, and `object_attr_setdouble_array()`.

Here is the call graph for this function:



### 38.69.2.24 jit\_attr\_setfloat()

```
t_jit_err jit_attr_setfloat (
    void * x,
    t_symbol * s,
    t_atom_float c )
```

Sets attribute value as a floating point value.

#### Parameters

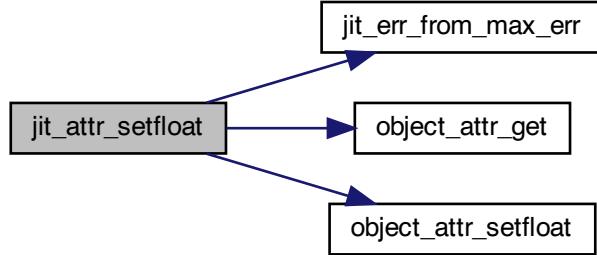
x	object pointer
s	attribute name
c	value

#### Returns

t\_jit\_err error code.

References jit\_err\_from\_max\_err(), object\_attr\_get(), and object\_attr\_setfloat().

Here is the call graph for this function:



### 38.69.2.25 jit\_attr\_setfloat\_array()

```
t_jit_err jit_attr_setfloat_array (
    void * x,
    t_symbol * s,
    long count,
    float * vals )
```

Sets attribute value as an array of floating point values.

#### Parameters

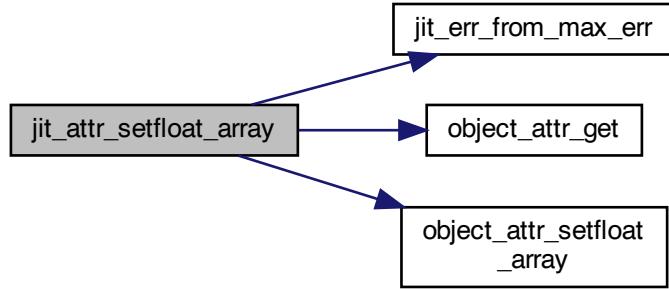
<i>x</i>	object pointer
<i>s</i>	attribute name
<i>count</i>	number of values
<i>vals</i>	pointer to values

#### Returns

`t_jit_err` error code.

References `jit_err_from_max_err()`, `object_attr_get()`, and `object_attr_setfloat_array()`.

Here is the call graph for this function:



### 38.69.2.26 jit\_attr\_setlong()

```
t_jit_err jit_attr_setlong (
    void * x,
    t_symbol * s,
    t_atom_long c )
```

Sets attribute value as a long integer value.

#### Parameters

x	object pointer
s	attribute name
c	value

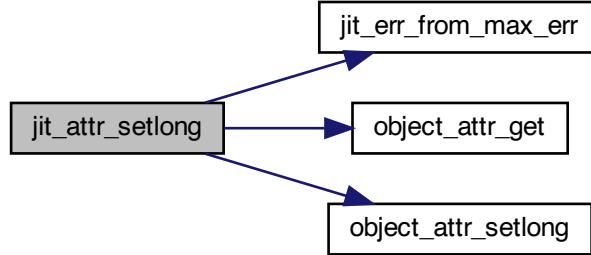
#### Returns

t\_jit\_err error code.

References jit\_err\_from\_max\_err(), object\_attr\_get(), and object\_attr\_setlong().

Referenced by jit\_mop\_input\_nolink(), jit\_mop\_output\_nolink(), jit\_mop\_single\_plane\_count(), and max\_jit\_mop\_matrix\_args().

Here is the call graph for this function:



### 38.69.2.27 jit\_attr\_setlong\_array()

```
t_jit_err jit_attr_setlong_array (
    void * x,
    t_symbol * s,
    long count,
    t_atom_long * vals )
```

Sets attribute value as an array of long integer values.

#### Parameters

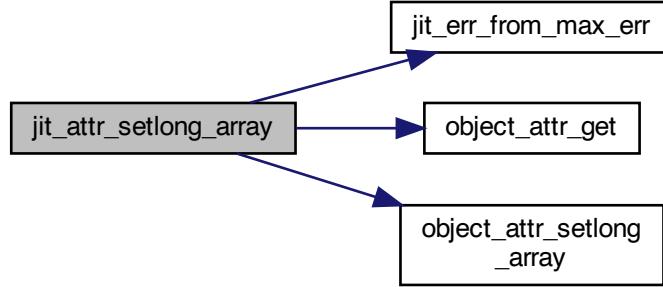
<i>x</i>	object pointer
<i>s</i>	attribute name
<i>count</i>	number of values
<i>vals</i>	pointer to values

#### Returns

*t\_jit\_err* error code.

References `jit_err_from_max_err()`, `object_attr_get()`, and `object_attr_setlong_array()`.

Here is the call graph for this function:



### 38.69.2.28 jit\_attr\_setsym()

```
t_jit_err jit_attr_setsym (
    void * x,
    t_symbol * s,
    t_symbol * c )
```

Sets attribute value as a symbol value.

#### Parameters

x	object pointer
s	attribute name
c	value

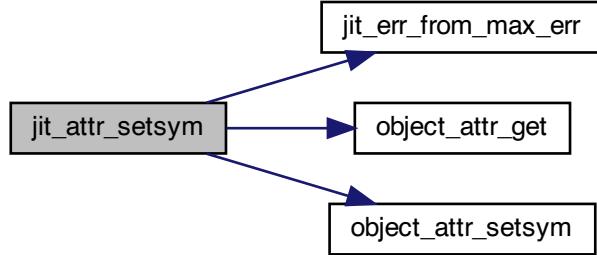
#### Returns

t\_jit\_err error code.

References jit\_err\_from\_max\_err(), object\_attr\_get(), and object\_attr\_setsym().

Referenced by jit\_matrix\_exprfill(), jit\_matrix\_op(), jit\_mop\_new(), max\_jit\_mop\_free(), max\_jit\_mop\_inputs(), max\_jit\_mop\_jit\_matrix(), max\_jit\_mop\_notify(), and max\_jit\_mop\_outputs().

Here is the call graph for this function:



### 38.69.2.29 jit\_attr\_setsym\_array()

```
t_jit_err jit_attr_setsym_array (
    void * x,
    t_symbol * s,
    long count,
    t_symbol ** vals )
```

Sets attribute value as an array of symbol values.

#### Parameters

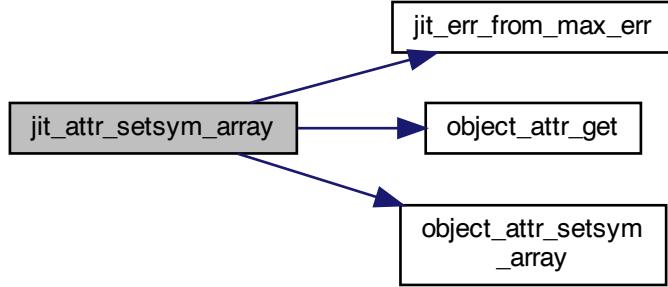
<i>x</i>	object pointer
<i>s</i>	attribute name
<i>count</i>	number of values
<i>vals</i>	pointer to values

#### Returns

`t_jit_err` error code.

References `jit_err_from_max_err()`, `object_attr_get()`, and `object_attr_setsym_array()`.

Here is the call graph for this function:



### 38.69.2.30 jit\_attr\_symcompare()

```
long jit_attr_symcompare (
    void * x,
    t_symbol * name )
```

Compares symbol name with name provided.

#### Parameters

<i>x</i>	attribute object pointer
<i>name</i>	attribute name

#### Returns

1 if equal, 0 if not equal

References `_jit_sym_getname`, and `object_method()`.

Referenced by `max_jit_obex_attr_get()`, and `max_jit_obex_attr_set()`.

Here is the call graph for this function:



### 38.69.2.31 jit\_attr\_usercanget()

```
t_atom_long jit_attr_usercanget (
    t_jit_attr * x )
```

Retrieves attribute user gettable flag.

#### Parameters

x	attribute object pointer
---	--------------------------

#### Returns

user gettable flag

#### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of any attribute obejct.

References `t_jit_attr::flags`, and `JIT_ATTR_GET_OPAQUE_USER`.

### 38.69.2.32 jit\_attr\_usercanset()

```
t_atom_long jit_attr_usercanset (
    t_jit_attr * x )
```

Retrieves attribute user settable flag.

**Parameters**

x	attribute object pointer
---	--------------------------

**Returns**

user settable flag

**Warning**

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of any attribute obejct.

References [t\\_jit\\_attr::flags](#), and [JIT\\_ATTR\\_SET\\_OPAQUE\\_USER](#).

**38.69.2.33 jit\_attribute\_new()**

```
t_jit_object * jit_attribute_new (
    char * name,
    t_symbol * type,
    long flags,
    method mget,
    method mset )
```

Constructs instance of [t\\_jit\\_attribute](#).

**Parameters**

<i>name</i>	attribute name
<i>type</i>	data type
<i>flags</i>	privacy flags
<i>mget</i>	getter method
<i>mset</i>	setter method

**Returns**

[t\\_jit\\_attribute](#) object pointer

**Warning**

This function is not exported, but is provided for reference when calling via jit\_object\_new.

References [t\\_jit\\_attribute::data](#), [t\\_jit\\_attribute::filterget](#), [t\\_jit\\_attribute::filterset](#), [t\\_jit\\_attribute::flags](#), [gensym\(\)](#), [t\\_jit\\_attribute::get](#), [t\\_jit\\_attribute::name](#), [t\\_jit\\_attribute::reserved](#), [t\\_jit\\_attribute::set](#), [t\\_jit\\_attribute::size](#), and [t\\_jit\\_attribute::type](#).

Here is the call graph for this function:



## 38.70 Binary Module

Collaboration diagram for Binary Module:



### Functions

- `t_jit_err jit_bin_read_header (t_filehandle fh, t_uint32 *version, t_int32 *filesize)`  
*Reads the header of a JXF binary file.*
- `t_jit_err jit_bin_read_chunk_info (t_filehandle fh, t_uint32 *ckid, t_int32 *cksize)`  
*Reads the info of a chunk from a JXF binary file.*
- `t_jit_err jit_bin_write_header (t_filehandle fh, t_int32 filesize)`  
*Writes the header of a JXF binary file.*
- `t_jit_err jit_bin_read_matrix (t_filehandle fh, void *matrix)`  
*Reads matrix data from a JXF binary file.*
- `t_jit_err jit_bin_write_matrix (t_filehandle fh, void *matrix)`  
*Writes a matrix to a JXF binary file.*

#### 38.70.1 Detailed Description

#### 38.70.2 Function Documentation

### 38.70.2.1 jit\_bin\_read\_chunk\_info()

```
t_jit_err jit_bin_read_chunk_info (
    t_filehandle fh,
    t_uint32 * ckid,
    t_int32 * cksize )
```

Reads the info of a chunk from a JXF binary file.

#### Parameters

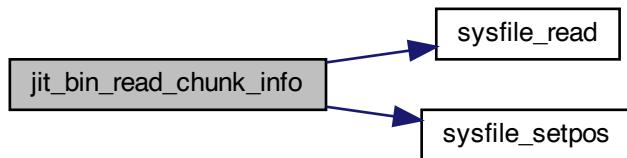
<i>fh</i>	t_filehandle file handle
<i>ckid</i>	chunk ID (ie JIT_BIN_CHUNK_CONTAINER, JIT_BIN_CHUNK_MATRIX)
<i>cksizes</i>	the size of the chunk

#### Returns

t\_jit\_err error code.

References SYSFILE\_FROMMARK, sysfile\_read(), and sysfile\_setpos().

Here is the call graph for this function:



### 38.70.2.2 jit\_bin\_read\_header()

```
t_jit_err jit_bin_read_header (
    t_filehandle fh,
    t_uint32 * version,
    t_int32 * filesize )
```

Reads the header of a JXF binary file.

**Parameters**

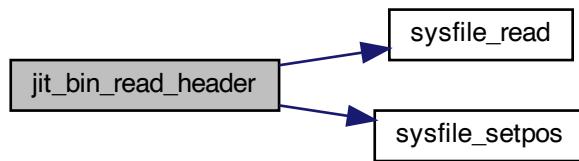
<i>fh</i>	t_filehandle file handle
<i>version</i>	version of the binary file format (ie JIT_BIN_VERSION_1)
<i>filesize</i>	the size of the file

**Returns**

t\_jit\_err error code.

References SYSFILE\_FROMSTART, sysfile\_read(), and sysfile\_setpos().

Here is the call graph for this function:

**38.70.2.3 jit\_bin\_read\_matrix()**

```
t_jit_err jit_bin_read_matrix (
    t_filehandle fh,
    void * matrix )
```

Reads matrix data from a JXF binary file.

**Parameters**

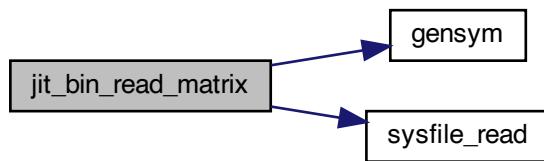
<i>fh</i>	t_filehandle file handle
<i>matrix</i>	the matrix data

**Returns**

`t_jit_err` error code.

References `_jit_sym_char`, `_jit_sym_float32`, `_jit_sym_float64`, `_jit_sym_getdata`, `_jit_sym_getinfo`, `_jit_sym_lock`, `_jit_sym_long`, `_jit_sym_setinfo`, `t_jit_matrix_info::dim`, `t_jit_matrix_info::dimcount`, `t_jit_matrix_info::flags`, `gensym()`, `JIT_MATRIX_MAX_DIMCOUNT`, `t_jit_matrix_info::planecount`, `sysfile_read()`, and `t_jit_matrix_info::type`.

Here is the call graph for this function:

**38.70.2.4 jit\_bin\_write\_header()**

```
t_jit_err jit_bin_write_header (
    t_filehandle fh,
    t_int32 filesize )
```

Writes the header of a JXF binary file.

**Parameters**

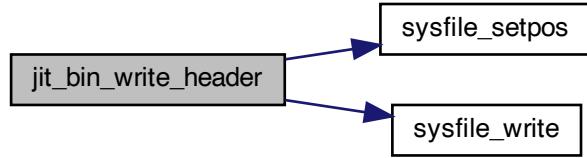
<code>fh</code>	<code>t_filehandle</code> file handle
<code>filesize</code>	the size of the file

**Returns**

`t_jit_err` error code.

References `SYSFILE_FROMSTART`, `sysfile_setpos()`, and `sysfile_write()`.

Here is the call graph for this function:



### 38.70.2.5 jit\_bin\_write\_matrix()

```
t_jit_err jit_bin_write_matrix (
    t_filehandle fh,
    void * matrix )
```

Writes a matrix to a JXF binary file.

#### Parameters

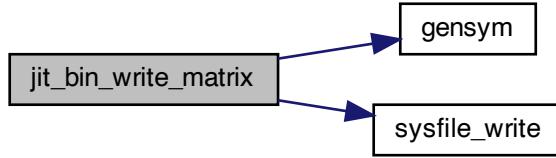
<i>fh</i>	t_filehandle file handle
<i>matrix</i>	the matrix data

#### Returns

t\_jit\_err error code.

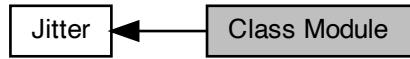
References \_jit\_sym\_char, \_jit\_sym\_float32, \_jit\_sym\_float64, \_jit\_sym\_long, t\_jit\_matrix\_info::dim, t\_jit\_matrix\_info::dimcount, gensym(), JIT\_MATRIX\_MAX\_DIMCOUNT, t\_jit\_matrix\_info::planecount, sysfile\_write(), and t\_jit\_matrix\_info::type.

Here is the call graph for this function:



## 38.71 Class Module

Collaboration diagram for Class Module:



## Functions

- `t_max_err class_copy (t_symbol *src_name_space, t_symbol *src_classname, t_symbol *dst_name_space, t_symbol *dst_classname)`

*Duplicates a previously registered object class, and registers a copy of this class.*
- `void * jit_class_new (C74_CONST char *name, method mnnew, method mfree, long size,...)`

*Creates a new class with the name specified by the name argument.*
- `t_jit_err jit_class_addmethod (void *c, method m, const char *name,...)`

*Adds a named method to a class.*
- `t_jit_err jit_class_addattr (void *c, t_jit_object *attr)`

*Adds an attribute to a class.*
- `t_jit_err jit_class_addadornment (void *c, t_jit_object *o)`

*Adds an adornment to a class.*
- `t_jit_err jit_class_addinterface (void *c, void *interfaceclass, long byteoffset, long flags)`

*Adds an interface to a class.*
- `void * jit_class_adornment_get (void *c, t_symbol *classname)`

*Retrieves an adornment from a class.*
- `t_jit_err jit_class_free (void *c)`

- `t_symbol * jit_class_nameget (void *c)`  
*Retrieves the name of a class.*
- `long jit_class_symcompare (void *c, t_symbol *name)`  
*Compares name of class with the name provided.*
- `t_jit_err jit_class_register (void *c)`  
*Registers class in the class registry.*
- `method jit_class_method (void *c, t_symbol *methodname)`  
*Retrieves method function pointer for named method.*
- `t_messlist * jit_class_mess (t_jit_class *c, t_symbol *methodname)`  
*Retrieves messlist entry for named method.*
- `void * jit_class_attr_get (void *c, t_symbol *attrname)`  
*Retrieves attribute pointer associated with name provided.*
- `void * jit_class_findbyname (t_symbol *classname)`  
*Retrieves class pointer associated with name provided.*
- `t_jit_err jit_class_addtypedwrapper (void *c, method m, char *name,...)`  
*Adds a typed wrapper method to a class.*
- `t_messlist * jit_class_typedwrapper_get (void *c, t_symbol *s)`  
*Retrieves typed wrapper messlist pointer associated with name provided.*
- `t_jit_err jit_class_method_addargsafe (void *c, char *argname, char *methodname)`  
*Marks a method as safe to call as an attribute style argument.*
- `t_symbol * jit_class_method_argsafe_get (void *c, t_symbol *s)`  
*Checks to see if symbol is safe to call as an attribute style argument.*

### 38.71.1 Detailed Description

### 38.71.2 Function Documentation

#### 38.71.2.1 `class_copy()`

```
t_max_err class_copy (
    t_symbol * src_name_space,
    t_symbol * src_classname,
    t_symbol * dst_name_space,
    t_symbol * dst_classname )
```

Duplicates a previously registered object class, and registers a copy of this class.

##### Parameters

<code>src_name_space</code>	The source class's name space.
<code>src_classname</code>	The source class's class name.
<code>dst_name_space</code>	The copied class's name space.
<code>dst_classname</code>	The copied class's class name.

**Returns**

This function returns the error code MAX\_ERR\_NONE if successful, or one of the other error codes defined in "ext\_obex.h" if unsuccessful.

**38.71.2.2 jit\_class\_addadornment()**

```
t_jit_err jit_class_addadornment (
    void * c,
    t_jit_object * o )
```

Adds an adornment to a class.

Adornments provide additional state and behavior to a class. This is most commonly used for the jit\_mop adornment.

**Parameters**

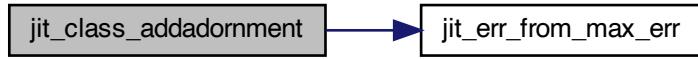
c	class pointer
o	object to use as adornment

**Returns**

t\_jit\_err error code

References jit\_err\_from\_max\_err().

Here is the call graph for this function:

**38.71.2.3 jit\_class\_addattr()**

```
t_jit_err jit_class_addattr (
    void * c,
    t_jit_object * attr )
```

Adds an attribute to a class.

**Parameters**

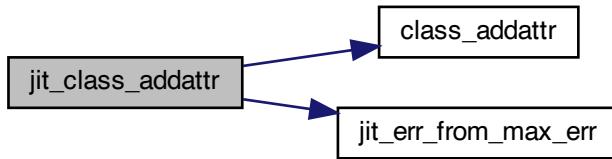
<i>c</i>	class pointer
<i>attr</i>	attribute object

**Returns**

`t_jit_err` error code

References `class_addattr()`, and `jit_err_from_max_err()`.

Here is the call graph for this function:

**38.71.2.4 `jit_class_addinterface()`**

```
t_jit_err jit_class_addinterface (
    void * c,
    void * interfaceclass,
    long byteoffset,
    long flags )
```

Adds an interface to a class.

Automatically expose methods and attributes of an interface class to a classes. Can also be used for class containers or subclassing behavior. If method or attribute is present in interface class prior to this call, the interface class' method or attribute will not be added. Use a nonzero byteoffset to contained class' object pointer in struct for container class. Use byte offset of zero for interface or subclassing behavior.

**Parameters**

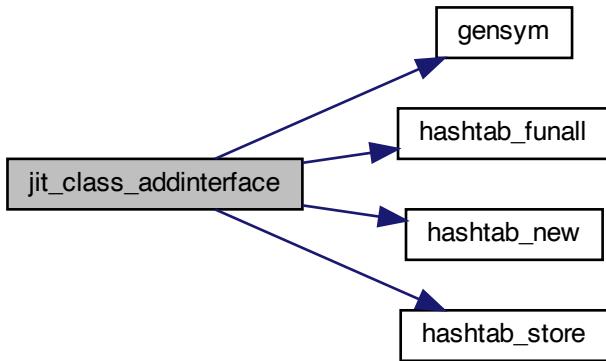
<i>c</i>	class pointer
<i>interfaceclass</i>	interface class pointer
<i>byteoffset</i>	byte offset (if for a contained object)
<i>flags</i>	reserved for future use

**Returns**

t\_jit\_err error code

References gensym(), hashtable\_funall(), hashtable\_new(), and hashtable\_store().

Here is the call graph for this function:



### 38.71.2.5 jit\_class\_addmethod()

```
t_jit_err jit_class_addmethod (
    void * c,
    method m,
    const char * name,
    ...
)
```

Adds a named method to a class.

**Parameters**

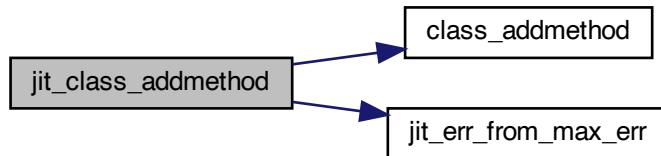
<i>c</i>	class pointer
<i>m</i>	function called when method is invoked
<i>name</i>	method name
...	type signature for the method in the standard Max type list format (see Chapter 3 of the Writing Externals in Max document for more information)

**Returns**

`t_jit_err` error code

References `class_addmethod()`, and `jit_err_from_max_err()`.

Here is the call graph for this function:

**38.71.2.6 jit\_class\_addtypedwrapper()**

```
t_jit_err jit_class_addtypedwrapper (
    void * c,
    method m,
    char * name,
    ...
)
```

Adds a typed wrapper method to a class.

Typed wrappers typically are used when there is an existing private, untyped method defined for a Jitter class, but it is desirable to expose the method to language bindings which require a typed interface—e.g. Java or JavaScript.

**Parameters**

<i>c</i>	class pointer
<i>m</i>	function called when method is invoked
<i>name</i>	method name
...	type signature for the method in the standard Max type list format (see Chapter 3 of the Writing External in Max document for more information)

**Returns**

`t_jit_err` error code

References `jit_err_from_max_err()`.

Here is the call graph for this function:



### 38.71.2.7 jit\_class\_adornment\_get()

```
void* jit_class_adornment_get (
    void * c,
    t_symbol * classname )
```

Retrieves an adornment from a class.

Adornments provide additional state and behavior to a class. This is most commonly used for the jit\_mop adornment.

#### Parameters

c	class pointer
classname	classname of adornment to retrieve

#### Returns

t\_jit\_err error code

Referenced by max\_jit\_classex\_mop\_mproc(), max\_jit\_classex\_mop\_wrap(), and max\_jit\_mop\_setup().

### 38.71.2.8 jit\_class\_attr\_get()

```
void* jit_class_attr_get (
    void * c,
    t_symbol * attrname )
```

Retrieves attribute pointer associated with name provided.

**Parameters**

<i>c</i>	class pointer
<i>attrname</i>	attribute name

**Returns**

attribute object pointer

**38.71.2.9 jit\_class\_findbyname()**

```
void* jit_class_findbyname (
    t_symbol * classname )
```

Retrieves class pointer associated with name provided.

**Parameters**

<i>classname</i>	class name
------------------	------------

**Returns**

class pointer

References [class\\_findbyname\(\)](#).

Here is the call graph for this function:

**38.71.2.10 jit\_class\_free()**

```
t_jit_err jit_class_free (
    void * c )
```

Frees a class.

**Warning**

This function is not typically used outside of jitlib.

**Parameters**

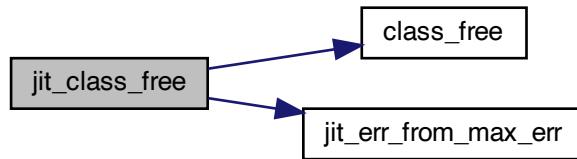
<i>c</i>	class pointer
----------	---------------

**Returns**

*t\_jit\_err* error code

References `class_free()`, and `jit_err_from_max_err()`.

Here is the call graph for this function:



### 38.71.2.11 jit\_class\_mess()

```
t_messlist* jit_class_mess (
    t_jit_class * c,
    t_symbol * methodname )
```

Retrieves messlist entry for named method.

**Parameters**

<i>c</i>	class pointer
<i>methodname</i>	method name

**Returns**

*t\_messlist* pointer.

### 38.71.2.12 jit\_class\_method()

```
method jit_class_method (
    void * c,
    t_symbol * methodname )
```

Retrieves method function pointer for named method.

#### Parameters

<i>c</i>	class pointer
<i>methodname</i>	method name

#### Returns

method function pointer.

### 38.71.2.13 jit\_class\_method\_addargsafe()

```
t_jit_err jit_class_method_addargsafe (
    void * c,
    char * argname,
    char * methodname )
```

Marks a method as safe to call as an attribute style argument.

#### Warning

It is important that no argument settable method causes any output into the patcher, or else it could lead to a crash, or other undesired behavior.

#### Parameters

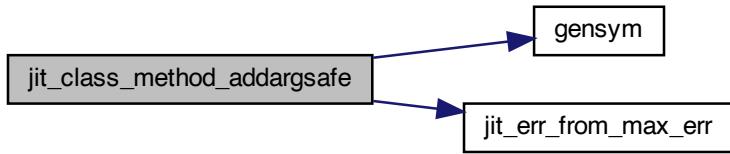
<i>c</i>	class pointer
<i>argname</i>	name as used via argument
<i>methodname</i>	name of method to map the argument name to

#### Returns

t\_jit\_err error code

References gensym(), and jit\_err\_from\_max\_err().

Here is the call graph for this function:



#### 38.71.2.14 jit\_class\_method\_argsafe\_get()

```
t_symbol* jit_class_method_argsafe_get (
    void * c,
    t_symbol * s )
```

Checks to see if symbol is safe to call as an attribute style argument.

##### Parameters

<i>c</i>	class pointer
<i>s</i>	name as used via argument

##### Returns

If successful, name of method to map the argument name to. Otherwise, NULL.

References gensym(), and t\_symbol::s\_name.

Referenced by jit\_object\_method\_argsafe\_get().

Here is the call graph for this function:



### 38.71.2.15 jit\_class\_nameget()

```
t_symbol* jit_class_nameget (
    void * c )
```

Retrieves the name of a class.

#### Parameters

c	class pointer
---	---------------

#### Returns

`t_symbol` pointer containing name of class

References `class_nameget()`.

Referenced by `jit_class_symcompare()`.

Here is the call graph for this function:



### 38.71.2.16 jit\_class\_new()

```
void* jit_class_new (
    C74_CONST char * name,
    method mnew,
    method mfree,
    long size,
    ... )
```

Creates a new class with the name specified by the name argument.

### Parameters

<i>name</i>	class name
<i>mnew</i>	class constructor
<i>mfree</i>	class destructor
<i>size</i>	object struct size in bytes
...	type signature for the constructor in the standard Max type list format (see Chapter 3 of the Writing Externals in Max document for more information)

### Warning

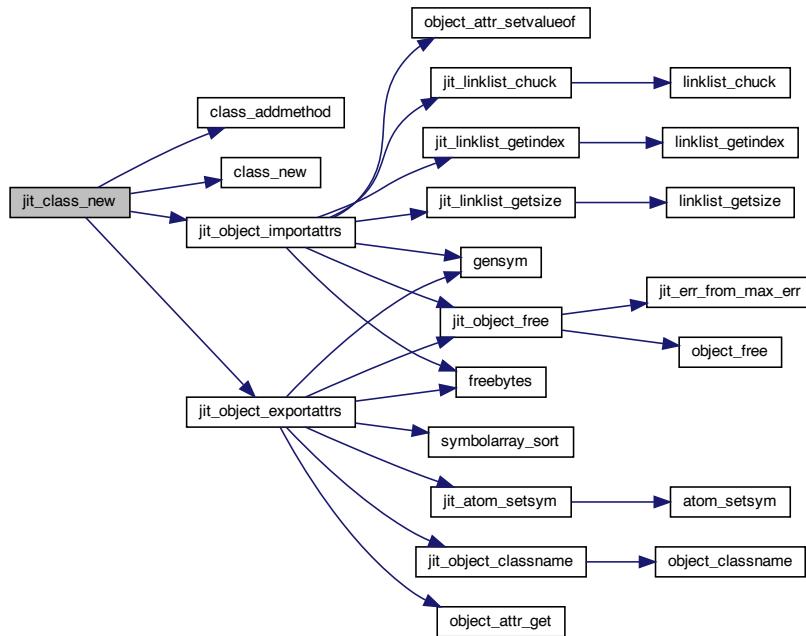
In order for the Jitter class to be exposed to JavaScript and Java, it is important that the constructor is typed, even if no arguments are provided—i.e. do not use the older strategy of defining Jitter constructors as private and untyped with A\_CANT.

### Returns

class pointer to be used in other class functions

References A\_CANT, A\_GIMME, class\_addmethod(), class\_new(), jit\_object\_exportatrs(), and jit\_object\_importatrs().

Here is the call graph for this function:



### 38.71.2.17 jit\_class\_register()

```
t_jit_err jit_class_register (
    void * c )
```

Registers class in the class registry.

#### Parameters

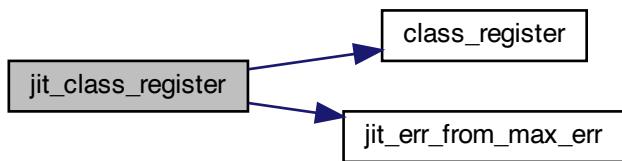
<i>c</i>	class pointer
----------	---------------

#### Returns

*t\_jit\_err* error code

References `class_register()`, and `jit_err_from_max_err()`.

Here is the call graph for this function:



### 38.71.2.18 jit\_class\_symcompare()

```
long jit_class_symcompare (
    void * c,
    t_symbol * name )
```

Compares name of class with the name provided.

#### Parameters

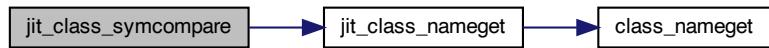
<i>c</i>	class pointer
<i>name</i>	name to compare with class name

**Returns**

1 if equal, 0 if not equal

References jit\_class\_nameget().

Here is the call graph for this function:

**38.71.2.19 jit\_class\_typedwrapper\_get()**

```
t_messlist* jit_class_typedwrapper_get (
    void * c,
    t_symbol * s )
```

Retrieves typed wrapper messlist pointer associated with name provided.

**Parameters**

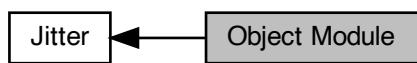
<i>c</i>	class pointer
<i>s</i>	name

**Returns**

*t\_messlist* pointer

**38.72 Object Module**

Collaboration diagram for Object Module:



## Functions

- long `jit_object_classname_compare` (void \*x, `t_symbol` \*name)
 

*Compares object's class name with the name provided.*
- `t_symbol * jit_object_method_argsafe_get` (void \*x, `t_symbol` \*s)
 

*Checks to see if symbol is safe to call as an attribute style argument.*
- void \* `jit_object_new_imp` (void \*cn, void \*p1, void \*p2, void \*p3, void \*p4, void \*p5, void \*p6, void \*p7, void \*p8, void \*dummy)
 

*Instantiates an object specified by class name.*
- void \* `jit_object_method_imp` (void \*x, void \*s, void \*p1, void \*p2, void \*p3, void \*p4, void \*p5, void \*p6, void \*p7, void \*p8)
 

*Calls an object method specified by method name.*
- void \* `jit_object_method_typed` (void \*x, `t_symbol` \*s, long ac, `t_atom` \*av, `t_atom` \*rv)
 

*Calls a typed object method specified by method name.*
- method `jit_object_getmethod` (void \*x, `t_symbol` \*s)
 

*Retrieves an object method specified by method name.*
- long `jit_object_attr_usercanset` (void \*x, `t_symbol` \*s)
 

*Determines if an object attribute is user settable.*
- long `jit_object_attr_usercanget` (void \*x, `t_symbol` \*s)
 

*Determines if an object attribute is user gettable.*
- void \* `jit_object_attr_get` (void \*x, `t_symbol` \*attrname)
 

*Retrieves an object's attribute pointer specified by attribute name.*
- `t_jit_err jit_object_free` (void \*x)
 

*Frees an object.*
- `t_symbol * jit_object_classname` (void \*x)
 

*Retrieves an object's class name.*
- void \* `jit_object_class` (void \*x)
 

*Retrieves an object's class pointer.*
- void \* `jit_object_register` (void \*x, `t_symbol` \*s)
 

*Registers an object in the named object registry.*
- `t_jit_err jit_object_unregister` (void \*x)
 

*Unregisters an object from the named object registry.*
- void \* `jit_object_findregistered` (`t_symbol` \*s)
 

*Retrieves a registered object associated with name.*
- `t_symbol * jit_object_findregisteredbyptr` (void \*x)
 

*Retrieves a registered object's name.*
- void \* `jit_object_attach` (`t_symbol` \*s, void \*x)
 

*Attaches an object as a client of a named server object for notification.*
- `t_jit_err jit_object_detach` (`t_symbol` \*s, void \*x)
 

*Detaches a client object from a named server object.*
- `t_jit_err jit_object_notify` (void \*x, `t_symbol` \*s, void \*data)
 

*Notifies all client objects for a named server object.*
- `t_jit_err jit_object_importatrs` (void \*x, `t_symbol` \*s, long argc, `t_atom` \*argv)
 

*Imports object attributes from an XML file.*
- `t_jit_err jit_object_exportatrs` (void \*x, `t_symbol` \*s, long argc, `t_atom` \*argv)
 

*Exports object attributes to an XML file.*
- `t_jit_err jit_object_exportsummary` (void \*x, `t_symbol` \*s, long argc, `t_atom` \*argv)
 

*Exports object summary to an XML file.*

### 38.72.1 Detailed Description

### 38.72.2 Function Documentation

#### 38.72.2.1 jit\_object\_attach()

```
void* jit_object_attach (
    t_symbol * s,
    void * x )
```

Attaches an object as a client of a named server object for notification.

##### Parameters

s	name of server object
x	client object pointer

##### Returns

If successful, server object pointer. Otherwise NULL.

References object\_attach().

Referenced by max\_jit\_mop\_inputs(), max\_jit\_mop\_notify(), and max\_jit\_mop\_outputs().

Here is the call graph for this function:



#### 38.72.2.2 jit\_object\_attr\_get()

```
void* jit_object_attr_get (
    void * x,
    t_symbol * attrname )
```

Retrieves an object's attribute pointer specified by attribute name.

**Parameters**

<i>x</i>	object pointer
<i>attrname</i>	attribute name

**Returns**

attribute object pointer

References `object_attr_get()`.

Here is the call graph for this function:

**38.72.2.3 jit\_object\_attr\_usercanget()**

```
long jit_object_attr_usercanget (
    void * x,
    t_symbol * s )
```

Determines if an object attribute is user gettable.

**Parameters**

<i>x</i>	object pointer
<i>s</i>	attribute name

**Returns**

1 if gettable, 0 if not gettable

References `object_attr_usercanget()`.

Here is the call graph for this function:



#### 38.72.2.4 jit\_object\_attr\_usercanset()

```
long jit_object_attr_usercanset (
    void * x,
    t_symbol * s )
```

Determines if an object attribute is user settable.

##### Parameters

x	object pointer
s	attribute name

##### Returns

1 if settable, 0 if not settable

References object\_attr\_usercanset().

Referenced by max\_jit\_attr\_args().

Here is the call graph for this function:



### 38.72.2.5 jit\_object\_class()

```
void* jit_object_class (
    void * x )
```

Retrieves an object's class pointer.

#### Parameters

x	object pointer
---	----------------

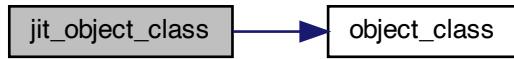
#### Returns

class pointer

References `object_class()`.

Referenced by `jit_object_method_argsafe_get()`, and `max_jit_mop_setup()`.

Here is the call graph for this function:



### 38.72.2.6 jit\_object\_classname()

```
t_symbol* jit_object_classname (
    void * x )
```

Retrieves an object's class name.

#### Parameters

x	object pointer
---	----------------

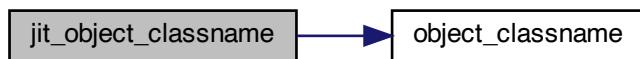
**Returns**

class name `t_symbol` pointer

References `object_classname()`.

Referenced by `jit_object_classname_compare()`, and `jit_object_exportattrs()`.

Here is the call graph for this function:



### 38.72.2.7 `jit_object_classname_compare()`

```
long jit_object_classname_compare (
    void * x,
    t_symbol * name )
```

Compares object's class name with the name provided.

**Parameters**

<code>x</code>	object pointer
<code>name</code>	name to compare with class name

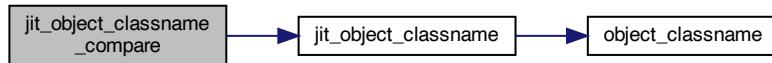
**Returns**

1 if equal, 0 if not equal

References `jit_object_classname()`.

Referenced by `max_jit_obex_adornment_get()`.

Here is the call graph for this function:



### 38.72.2.8 jit\_object\_detach()

```
t_jit_err jit_object_detach (
    t_symbol * s,
    void * x )
```

Detaches a client object from a named server object.

#### Parameters

s	name of server object
x	client object pointer

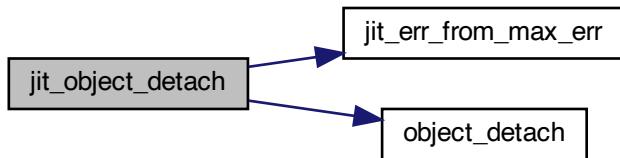
#### Returns

t\_jit\_err error code

References jit\_err\_from\_max\_err(), and object\_detach().

Referenced by max\_jit\_mop\_free().

Here is the call graph for this function:



### 38.72.2.9 jit\_object\_exportattrs()

```
t_jit_err jit_object_exportattrs (
    void * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Exports object attributes to an XML file.

#### Parameters

<i>x</i>	object pointer
<i>s</i>	ignored
<i>argc</i>	argument count
<i>argv</i>	argument vector

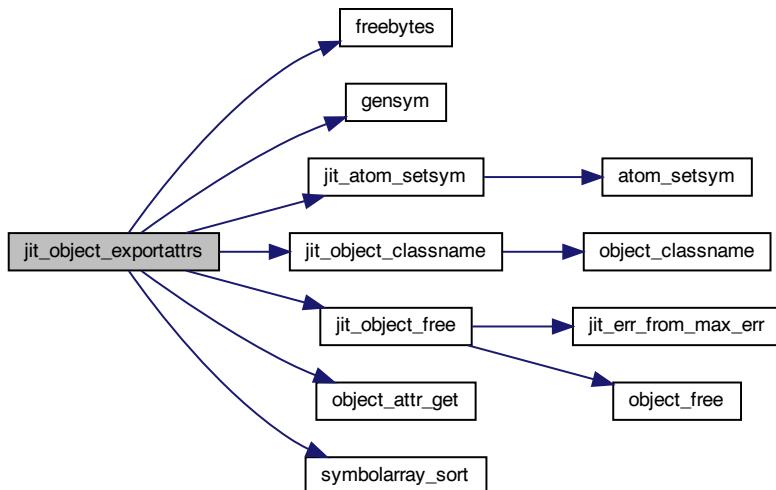
#### Returns

*t\_jit\_err* error code

References *\_jit\_sym\_getname*, *freebytes()*, *gensym()*, *jit\_atom\_setsym()*, *jit\_object\_classname()*, *jit\_object\_free()*, *object\_attr\_get()*, *t\_symbol::s\_name*, and *symbolarray\_sort()*.

Referenced by *jit\_class\_new()*, and *max\_jit\_classex\_standard\_wrap()*.

Here is the call graph for this function:



### 38.72.2.10 jit\_object\_exportsummary()

```
t_jit_err jit_object_exportsummary (
    void * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Exports object summary to an XML file.

#### Warning

Currently this function does nothing, but is reserved for future use.

#### Parameters

x	object pointer
s	ignored
argc	argument count
argv	argument vector

#### Returns

t\_jit\_err error code

Referenced by max\_jit\_classex\_standard\_wrap().

### 38.72.2.11 jit\_object\_findregistered()

```
void * jit_object_findregistered (
    t_symbol * s )
```

Retrieves a registered object associated with name.

#### Parameters

s	registered name
---	-----------------

#### Returns

If successful, object pointer. Otherwise NULL.

References object\_findregistered().

Referenced by jit\_matrix\_jit\_gl\_texture(), jit\_matrix\_op(), max\_jit\_mop\_jit\_matrix(), and max\_jit\_mop\_notify().

Here is the call graph for this function:



### 38.72.2.12 jit\_object\_findregisteredbyptr()

```
t_symbol* jit_object_findregisteredbyptr ( void * x )
```

Retrieves a registered object's name.

#### Parameters

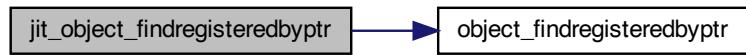
x	object pointer
---	----------------

#### Returns

If successful, [t\\_symbol](#) pointer name. Otherwise NULL.

References [object\\_findregisteredbyptr\(\)](#).

Here is the call graph for this function:



### 38.72.2.13 jit\_object\_free()

```
t_jit_err jit_object_free (
    void * x )
```

Frees an object.

#### Parameters

x	object pointer
---	----------------

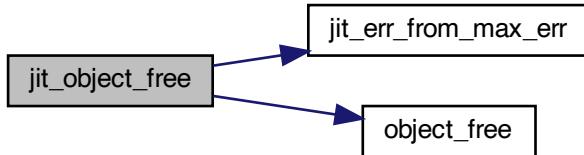
#### Returns

t\_jit\_err error code

References jit\_err\_from\_max\_err(), and object\_free().

Referenced by jit\_matrix\_exprfill(), jit\_matrix\_new(), jit\_matrix\_newcopy(), jit\_matrix\_op(), jit\_mop\_free(), jit\_mop\_new(), jit\_mop\_newcopy(), jit\_object\_exportattrs(), jit\_object\_importattrs(), max\_jit\_mop\_adapt\_matrix\_all(), max\_jit\_mop\_free(), and max\_jit\_obex\_free().

Here is the call graph for this function:



### 38.72.2.14 jit\_object\_getmethod()

```
method jit_object_getmethod (
    void * x,
    t_symbol * s )
```

Retrieves an object method specified by method name.

**Parameters**

<i>x</i>	object pointer
<i>s</i>	method name

**Returns**

method

References [object\\_getmethod\(\)](#).

Referenced by [jit\\_attr\\_filterget\(\)](#), and [jit\\_attr\\_filterset\(\)](#).

Here is the call graph for this function:

**38.72.2.15 jit\_object\_importattrs()**

```
t_jit_err jit_object_importattrs (
    void * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Imports object attributes from an XML file.

**Parameters**

<i>x</i>	object pointer
<i>s</i>	ignored
<i>argc</i>	argument count
<i>argv</i>	argument vector

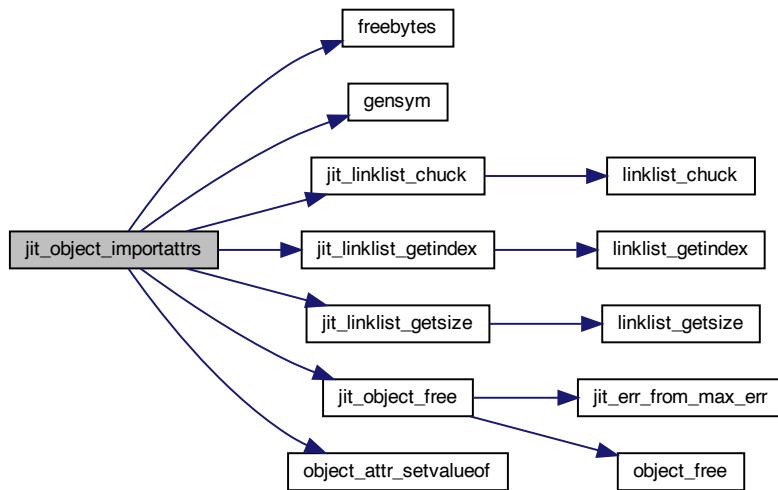
**Returns**

`t_jit_err` error code

References `_jit_sym_name`, `freebytes()`, `gensym()`, `jit_linklist_chuck()`, `jit_linklist_getindex()`, `jit_linklist_getsize()`, `jit_object_free()`, and `object_attr_setvalueof()`.

Referenced by `jit_class_new()`, and `max_jit_classex_standard_wrap()`.

Here is the call graph for this function:

**38.72.2.16 jit\_object\_method\_argsafe\_get()**

```
t_symbol* jit_object_method_argsafe_get (
    void * x,
    t_symbol * s )
```

Checks to see if symbol is safe to call as an attribute style argument.

**Parameters**

<code>x</code>	object pointer
<code>s</code>	name as used via argument

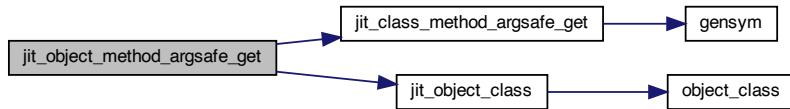
**Returns**

If successful, name of method to map the argument name to. Otherwise, NULL.

References jit\_class\_method\_argsafe\_get(), and jit\_object\_class().

Referenced by max\_jit\_attr\_args().

Here is the call graph for this function:

**38.72.2.17 jit\_object\_method\_imp()**

```

void* jit_object_method_imp (
    void * x,
    void * s,
    void * p1,
    void * p2,
    void * p3,
    void * p4,
    void * p5,
    void * p6,
    void * p7,
    void * p8 )
  
```

Calls an object method specified by method name.

This operation is untyped, and the contents of the stack following the method name argument are blindly passed to the method called.

**Parameters**

<i>x</i>	object pointer
<i>s</i>	method name
<i>p1</i>	untyped arguments passed on to the method
<i>p2</i>	untyped arguments passed on to the method
<i>p3</i>	untyped arguments passed on to the method
<i>p4</i>	untyped arguments passed on to the method
<i>p5</i>	untyped arguments passed on to the method
<i>p6</i>	untyped arguments passed on to the method
<i>p7</i>	untyped arguments passed on to the method
<i>p8</i>	untyped arguments passed on to the method

**Warning**

It is important to know any necessary arguments for untyped constructors such as those used by jit\_matrix or jit\_attr\_offset.

**Returns**

method dependent, but uses void \* as a super type.

**38.72.2.18 jit\_object\_method\_typed()**

```
void* jit_object_method_typed (
    void * x,
    t_symbol * s,
    long ac,
    t_atom * av,
    t_atom * rv )
```

Calls a typed object method specified by method name.

This operation only supports methods which are typed—i.e. it cannot be used to call private, untyped A\_CANT methods.

**Parameters**

<i>x</i>	object pointer
<i>s</i>	method name
<i>ac</i>	argument count
<i>av</i>	argument vector
<i>rv</i>	return value for A_GIMMEBACK methods

**Returns**

method dependent, but uses void \* as a super type.

References object\_method\_typed().

Here is the call graph for this function:



### 38.72.2.19 jit\_object\_new\_imp()

```
void* jit_object_new_imp (
    void * cn,
    void * p1,
    void * p2,
    void * p3,
    void * p4,
    void * p5,
    void * p6,
    void * p7,
    void * p8,
    void * dummy )
```

Instantiates an object specified by class name.

This function may be used to create instances of any Jitter object.

#### Parameters

<i>cn</i>	class name
<i>p1</i>	untyped arguments passed on to the constructor
<i>p2</i>	untyped arguments passed on to the constructor
<i>p3</i>	untyped arguments passed on to the constructor
<i>p4</i>	untyped arguments passed on to the constructor
<i>p5</i>	untyped arguments passed on to the constructor
<i>p6</i>	untyped arguments passed on to the constructor
<i>p7</i>	untyped arguments passed on to the constructor
<i>p8</i>	untyped arguments passed on to the constructor
<i>dummy</i>	unused

#### Warning

It is important to know any necessary arguments for untyped constructors such as those used by `jit_matrix` or `jit_attr_offset`.

#### Returns

If successful, a valid object pointer. Otherwise, NULL.

### 38.72.2.20 jit\_object\_notify()

```
t_jit_err jit_object_notify (
    void * x,
    t_symbol * s,
    void * data )
```

Notifies all client objects for a named server object.

#### Parameters

<i>x</i>	server object pointer
<i>s</i>	notification message
<i>data</i>	message specific data

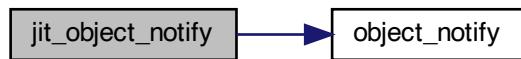
#### Returns

*t\_jit\_err* error code

References [object\\_notify\(\)](#).

Referenced by [jit\\_matrix\\_setinfo\(\)](#), and [jit\\_matrix\\_setinfo\\_ex\(\)](#).

Here is the call graph for this function:



### 38.72.2.21 jit\_object\_register()

```
void* jit_object_register (
    void * x,
    t_symbol * s )
```

Registers an object in the named object registry.

#### Parameters

<i>x</i>	object pointer
<i>s</i>	object name

**Returns**

object pointer

**Warning**

It is important to use the object pointer returned by jit\_object\_register, since if there is an existing object with the same name and class, it could free the input object and pass back a reference to the previously defined object.

References object\_register().

Referenced by max\_jit\_mop\_inputs(), max\_jit\_mop\_notify(), and max\_jit\_mop\_outputs().

Here is the call graph for this function:

**38.72.2.22 jit\_object\_unregister()**

```
t_jit_err jit_object_unregister (
    void * x )
```

Unregisters an object from the named object registry.

**Parameters**

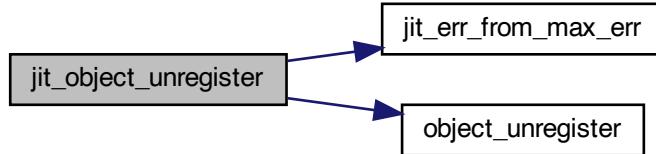
x	object pointer
---	----------------

**Returns**

t\_jit\_err error code

References jit\_err\_from\_max\_err(), and object\_unregister().

Here is the call graph for this function:



### 38.73 Miscellaneous Utility Module

Collaboration diagram for Miscellaneous Utility Module:



## Functions

- float [swapf32](#) (float f)  
*Byte swaps 32 bit floating point number.*
- double [swapf64](#) (double f)  
*Byte swaps 64 bit floating point number.*
- void [jit\\_global\\_critical\\_enter](#) (void)  
*Enters the global Jitter critical region.*
- void [jit\\_global\\_critical\\_exit](#) (void)  
*Exits the global Jitter critical region.*
- void [jit\\_error\\_sym](#) (void \*x, [t\\_symbol](#) \*s)  
*Sends symbol based error message to Max console (safe from all threads)*
- void [jit\\_error\\_code](#) (void \*x, [t\\_jit\\_err](#) v)  
*Sends error code based error message to Max console (safe from all threads)*
- void [jit\\_post\\_sym](#) (void \*x, [t\\_symbol](#) \*s)  
*Sends symbol based message to Max console (safe from all threads)*
- [t\\_jit\\_err jit\\_err\\_from\\_max\\_err](#) ([t\\_max\\_err](#) err)

*Converts Max style error codes to Jitter style error codes.*

- void `jit_rand_setseed` (long n)  
*Sets global random number generator seed.*
- long `jit_rand` (void)  
*Generates a random value as a signed long integer.*

### 38.73.1 Detailed Description

### 38.73.2 Function Documentation

#### 38.73.2.1 jit\_err\_from\_max\_err()

```
t_jit_err jit_err_from_max_err (
    t_max_err err )
```

Converts Max style error codes to Jitter style error codes.

Parameters

<code>err</code>	Max error code
------------------	----------------

Returns

`t_jit_err` error code

References `MAX_ERR_DUPLICATE`, `MAX_ERR_GENERIC`, `MAX_ERR_INVALID_PTR`, `MAX_ERR_NONE`, and `MAX_ERR_OUT_OF_MEM`.

Referenced by `jit_attr_setchar_array()`, `jit_attr_setdouble_array()`, `jit_attr_setfloat()`, `jit_attr_setfloat_array()`, `jit_attr_setlong()`, `jit_attr_setlong_array()`, `jit_attr_setsym()`, `jit_attr_setsym_array()`, `jit_class_addadornment()`, `jit_class_addattr()`, `jit_class_addmethod()`, `jit_class_adddtypedwrapper()`, `jit_class_free()`, `jit_class_method_addargsafe()`, `jit_class_register()`, `jit_object_detach()`, `jit_object_free()`, and `jit_object_unregister()`.

#### 38.73.2.2 jit\_error\_code()

```
void jit_error_code (
    void * x,
    t_jit_err v )
```

Sends error code based error message to Max console (safe from all threads)

**Parameters**

x	object pointer
v	error code

References defer().

Referenced by max\_jit\_attr\_getdump(), and max\_jit\_mop\_jit\_matrix().

Here is the call graph for this function:

**38.73.2.3 jit\_error\_sym()**

```
void jit_error_sym (
    void * x,
    t_symbol * s )
```

Sends symbol based error message to Max console (safe from all threads)

**Parameters**

x	object pointer
s	error message symbol

References defer().

Here is the call graph for this function:



#### 38.73.2.4 jit\_global\_critical\_enter()

```
void jit_global_critical_enter (
    void )
```

Enters the global Jitter critical region.

This function is useful for simple protection of thread sensitive operations. However, it may be too broad a lock, as it prevents any other operations that use the global critical region from working. For more localized control, I would suggest using either Max's systhread API or the platform specific locking mechanisms however, be sensitive to the possibility deadlock when locking code which calls code which may require the locking off unknown resources.

References [critical\\_enter\(\)](#).

Here is the call graph for this function:



#### 38.73.2.5 jit\_global\_critical\_exit()

```
void jit_global_critical_exit (
    void )
```

Exits the global Jitter critical region.

This function is useful for simple protection of thread sensitive operations. However, it may be too broad a lock, as it prevents any other operations that use the global critical region from working. For more localized control, I would suggest using either Max's systhread API or the platform specific locking mechanisms however, be sensitive to the possibility deadlock when locking code which calls code which may require the locking off unknown resources.

References [critical\\_exit\(\)](#).

Here is the call graph for this function:



### 38.73.2.6 jit\_post\_sym()

```
void jit_post_sym (
    void * x,
    t_symbol * s )
```

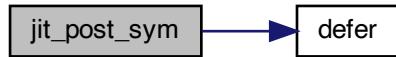
Sends symbol based message to Max console (safe from all threads)

#### Parameters

x	object pointer
s	message symbol

References defer().

Here is the call graph for this function:



### 38.73.2.7 jit\_rand()

```
long jit_rand (
    void )
```

Generates a random value as a signed long integer.

**Returns**

random value

**38.73.2.8 jit\_rand\_setseed()**

```
void jit_rand_setseed (
    long n )
```

Sets global random number generator seed.

**Parameters**

<i>n</i>	seed
----------	------

References systime\_ticks().

Here is the call graph for this function:

**38.73.2.9 swapf32()**

```
float swapf32 (
    float f )
```

Byte swaps 32 bit floating point number.

**Parameters**

<i>f</i>	input float
----------	-------------

**Returns**

byte swapped float

**38.73.2.10 swapf64()**

```
double swapf64 (
    double f )
```

Byte swaps 64 bit floating point number.

**Parameters**

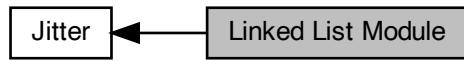
<i>f</i>	input double
----------	--------------

**Returns**

byte swapped double

**38.74 Linked List Module**

Collaboration diagram for Linked List Module:

**Functions**

- `void * jit_linklist_new (void)`  
*Constructs instance of t\_jit\_linklist.*
- `t_atom_long jit_linklist_getsize (t_jit_linklist *x)`  
*Retrieves the linked list size.*
- `void * jit_linklist_getindex (t_jit_linklist *x, long index)`  
*Retrieves the object at the specified list index.*
- `t_atom_long jit_linklist_objptr2index (t_jit_linklist *x, void *p)`  
*Retrieves the list index for an object pointer.*

- `t_atom_long jit_linklist_makearray (t_jit_linklist *x, void **a, long max)`  
*Flatten the linked list into an array.*
- `t_atom_long jit_linklist_insertindex (t_jit_linklist *x, void *o, long index)`  
*Insert object at specified index.*
- `t_atom_long jit_linklist_append (t_jit_linklist *x, void *o)`  
*Append object to the end of the linked list.*
- `t_atom_long jit_linklist_deleteindex (t_jit_linklist *x, long index)`  
*Delete object at specified index, freeing the object.*
- `t_atom_long jit_linklist_chuckindex (t_jit_linklist *x, long index)`  
*Remove object at specified index, without freeing the object.*
- `void jit_linklist_clear (t_jit_linklist *x)`  
*Clears the linked list, freeing all objects in list.*
- `void jit_linklist_chuck (t_jit_linklist *x)`  
*Removes all objects from the linked list, without freeing any objects in list.*
- `void jit_linklist_reverse (t_jit_linklist *x)`  
*Reverses the order of objects in the linked list.*
- `void jit_linklist_rotate (t_jit_linklist *x, long i)`  
*Rotates the order of objects in the linked list, by the specified number of indeces.*
- `void jit_linklist_shuffle (t_jit_linklist *x)`  
*Randomizes the order of objects in the linked list.*
- `void jit_linklist_swap (t_jit_linklist *x, long a, long b)`  
*Swap list location of the indeces specified.*
- `void jit_linklist_findfirst (t_jit_linklist *x, void **o, long cmpfn(void *, void *), void *cmpdata)`  
*Retrieves the first object that satisfies the comparison function.*
- `void jit_linklist_findall (t_jit_linklist *x, t_jit_linklist **out, long cmpfn(void *, void *), void *cmpdata)`  
*Retrieves a linked list of all objects that satisfy the comparison function.*
- `t_atom_long jit_linklist_findcount (t_jit_linklist *x, long cmpfn(void *, void *), void *cmpdata)`  
*Retrieves the number of objects that satisfy the comparison function.*
- `void jit_linklist_methodall (t_jit_linklist *x, t_symbol *s,...)`  
*Calls a method on all objects in linked list.*
- `void * jit_linklist_methodindex (t_jit_linklist *x, long i, t_symbol *s,...)`  
*Calls a method on the object at the specified index.*
- `void jit_linklist_sort (t_jit_linklist *x, long cmpfn(void *, void *))`  
*Sorts linked list based on the provided comparison function.*

### 38.74.1 Detailed Description

### 38.74.2 Function Documentation

#### 38.74.2.1 jit\_linklist\_append()

```
t_atom_long jit_linklist_append (
    t_jit_linklist * x,
    void * o )
```

Append object to the end of the linked list.

**Parameters**

<i>x</i>	t_jit_linklist object pointer
<i>o</i>	object pointer

**Returns**

new list length, or -1 if unsuccessful

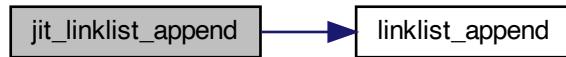
**Warning**

While exported, it is recommend to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_append().

Referenced by jit\_matrix\_op(), max\_jit\_classex\_addattr(), and max\_jit\_obex\_proxy\_new().

Here is the call graph for this function:

**38.74.2.2 jit\_linklist\_chuck()**

```
void jit_linklist_chuck (
    t_jit_linklist * x )
```

Removes all objects from the linked list, without freeing any objects in list.

To remove all objects from the linked list, freeing the objects, use the jit\_linklist\_clear method.

**Parameters**

<i>x</i>	t_jit_linklist object pointer
----------	-------------------------------

**Warning**

While exported, it is recommended to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_chuck().

Referenced by jit\_matrix\_op(), and jit\_object\_importattrs().

Here is the call graph for this function:



### 38.74.2.3 jit\_linklist\_chuckindex()

```
t_atom_long jit_linklist_chuckindex (
    t_jit_linklist * x,
    long index )
```

Remove object at specified index, without freeing the object.

This method will not free the object. To remove from the linked list and free the object, use the jit\_linklist\_deleteindex method.

**Parameters**

x	t_jit_linklist object pointer
index	index to remove (zero based)

**Returns**

index removed, or -1 if unsuccessful

### Warning

While exported, it is recommend to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_chuckindex().

Here is the call graph for this function:



### 38.74.2.4 jit\_linklist\_clear()

```
void jit_linklist_clear (
    t_jit_linklist * x )
```

Clears the linked list, freeing all objects in list.

To remove all elements from the linked list without freeing the objects, use the jit\_linklist\_chuck method.

#### Parameters

x	t_jit_linklist object pointer
---	-------------------------------

### Warning

While exported, it is recommend to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_clear().

Here is the call graph for this function:



### 38.74.2.5 jit\_linklist\_deleteindex()

```
t_atom_long jit_linklist_deleteindex (
    t_jit_linklist * x,
    long index )
```

Delete object at specified index, freeing the object.

To remove from the linked list without freeing the object, use the jit\_linklist\_chuckindex method.

#### Parameters

x	t_jit_linklist object pointer
index	index to delete (zero based)

#### Returns

index deleted, or -1 if unsuccessful

#### Warning

While exported, it is recommend to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_deleteindex().

Here is the call graph for this function:



### 38.74.2.6 jit\_linklist\_findall()

```
void jit_linklist_findall (
    t_jit_linklist * x,
    t_jit_linklist ** out,
    long cmpfnvoid *, void *,
    void * cmpdata )
```

Retrieves a linked list of all objects that satisfy the comparison function.

#### Parameters

<i>x</i>	t_jit_linklist object pointer
<i>out</i>	pointer to linked list containing all objects found found (set to NULL, if not found)
<i>cmpfn</i>	comparison function pointer (should returns 1 if object matches data, otherwise 0)
<i>cmpdata</i>	opaque data used in comparison function

#### Warning

While exported, it is recommend to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_findall().

Here is the call graph for this function:



### 38.74.2.7 jit\_linklist\_findcount()

```
t_atom_long jit_linklist_findcount (
    t_jit_linklist * x,
    long cmpfnvoid *, void *,
    void * cmpdata )
```

Retrieves the number of objects that satisfy the comparison function.

#### Parameters

<i>x</i>	<i>t_jit_linklist</i> object pointer
<i>cmpfn</i>	comparison function pointer (should returns 1 if object matches data, otherwise 0)
<i>cmpdata</i>	opaque data used in comparison function

#### Returns

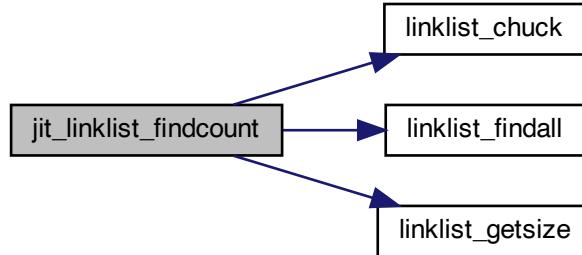
number object objects that satisfy the comparison function

#### Warning

While exported, it is recommend to use *jit\_object\_method* to call methods on an object when the object may not be an instance of *t\_jit\_linklist*, but instead an object that supports some portion of the *t\_jit\_linklist* interface. One instance where this is the case is inside of a MOP *matrix\_calc* method, where the arguments can be either an instance of *t\_jit\_linklist*, or *t\_jit\_matrix* which has a *getindex* method.

References *linklist\_chuck()*, *linklist\_findall()*, and *linklist\_getsize()*.

Here is the call graph for this function:



### 38.74.2.8 jit\_linklist\_findfirst()

```
void jit_linklist_findfirst (
    t_jit_linklist * x,
    void ** o,
    long cmpfnvoid *, void *,
    void * cmpdata )
```

Retrieves the first object that satisfies the comparison function.

#### Parameters

<i>x</i>	t_jit_linklist object pointer
<i>o</i>	pointer to object pointer found (set to NULL, if not found)
<i>cmpfn</i>	comparison function pointer (should returns 1 if object matches data, otherwise 0)
<i>cmpdata</i>	opaque data used in comparison function

#### Warning

While exported, it is recommended to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_findfirst().

Referenced by max\_jit\_obex\_attr\_get(), and max\_jit\_obex\_attr\_set().

Here is the call graph for this function:



### 38.74.2.9 jit\_linklist\_getindex()

```
void* jit_linklist_getindex (
    t_jit_linklist * x,
    long index )
```

Retrieves the object at the specified list index.

#### Parameters

x	t_jit_linklist object pointer
index	list index ()

#### Returns

object pointer

#### Warning

While exported, it is recommend to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_getindex().

Referenced by jit\_object\_importattrs().

Here is the call graph for this function:



### 38.74.2.10 jit\_linklist\_getsize()

```
t_atom_long jit_linklist_getsize (
    t_jit_linklist * x )
```

Retrieves the linked list size.

#### Parameters

x	t_jit_linklist object pointer
---	-------------------------------

#### Returns

linked list size

#### Warning

While exported, it is recommend to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_getsize().

Referenced by jit\_object\_importattrs().

Here is the call graph for this function:



### 38.74.2.11 jit\_linklist\_insertindex()

```
t_atom_long jit_linklist_insertindex (
    t_jit_linklist * x,
    void * o,
    long index )
```

Insert object at specified index.

#### Parameters

<i>x</i>	<code>t_jit_linklist</code> object pointer
<i>o</i>	object pointer
<i>index</i>	index (zero based)

#### Returns

`index` inserted at, or -1 if unsuccessful

#### Warning

While exported, it is recommend to use `jit_object_method` to call methods on an object when the object may not be an instance of `t_jit_linklist`, but instead an object that supports some portion of the `t_jit_linklist` interface. One instance where this is the case is inside of a MOP `matrix_calc` method, where the arguments can be either an instance of `t_jit_linklist`, or `t_jit_matrix` which has a `getindex` method.

References `linklist_insertindex()`.

Here is the call graph for this function:



### 38.74.2.12 jit\_linklist\_makearray()

```
t_atom_long jit_linklist_makearray (
    t_jit_linklist * x,
    void ** a,
    long max )
```

Flatten the linked list into an array.

**Parameters**

<i>x</i>	t_jit_linklist object pointer
<i>a</i>	array pointer
<i>max</i>	maximum array size

**Returns**

number of object pointers copied into array

**Warning**

While exported, it is recommended to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_makearray().

Here is the call graph for this function:

**38.74.2.13 jit\_linklist\_methodall()**

```
void jit_linklist_methodall (
    t_jit_linklist * x,
    t_symbol * s,
    ... )
```

Calls a method on all objects in linked list.

Equivalent to calling jit\_object\_method on the object at each index.

**Parameters**

<i>x</i>	t_jit_linklist object pointer
<i>s</i>	method name
...	untyped arguments

**Warning**

While exported, it is recommended to use `jit_object_method` to call methods on an object when the object may not be an instance of `t_jit_linklist`, but instead an object that supports some portion of the `t_jit_linklist` interface. One instance where this is the case is inside of a MOP `matrix_calc` method, where the arguments can be either an instance of `t_jit_linklist`, or `t_jit_matrix` which has a `getindex` method.

References `linklist_methodall()`.

Here is the call graph for this function:

**38.74.2.14 jit\_linklist\_methodindex()**

```
void* jit_linklist_methodindex (
    t_jit_linklist * x,
    long i,
    t_symbol * s,
    ...
)
```

Calls a method on the object at the specified index.

Equivalent to calling `jit_object_method` on the object.

**Parameters**

<code>x</code>	<code>t_jit_linklist</code> object pointer
<code>i</code>	index
<code>s</code>	method name
...	untyped arguments

**Returns**

method return value

### Warning

While exported, it is recommend to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_methodindex().

Here is the call graph for this function:



### 38.74.2.15 jit\_linklist\_new()

```
void* jit_linklist_new (
    void )
```

Constructs instance of t\_jit\_linklist.

#### Returns

t\_jit\_linklist object pointer

### Warning

While exported, it is recommend to use jit\_object\_new to construct a t\_jit\_linklist object.

References linklist\_new().

Referenced by jit\_matrix\_op(), max\_jit\_classex\_addattr(), and max\_jit\_obex\_proxy\_new().

Here is the call graph for this function:



**38.74.2.16 jit\_linklist\_objptr2index()**

```
t_atom_long jit_linklist_objptr2index (
    t_jit_linklist * x,
    void * p )
```

Retrieves the list index for an object pointer.

**Parameters**

x	t_jit_linklist object pointer
p	object pointer

**Returns**

object's list index (zero based), or -1 if not present

**Warning**

While exported, it is recommended to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_objptr2index().

Here is the call graph for this function:

**38.74.2.17 jit\_linklist\_reverse()**

```
void jit_linklist_reverse (
    t_jit_linklist * x )
```

Reverses the order of objects in the linked list.

**Parameters**

x	t_jit_linklist object pointer
---	-------------------------------

**Warning**

While exported, it is recommend to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_reverse().

Here is the call graph for this function:

**38.74.2.18 jit\_linklist\_rotate()**

```
void jit_linklist_rotate (
    t_jit_linklist * x,
    long i )
```

Rotates the order of objects in the linked list, by the specified number of indeces.

**Parameters**

x	t_jit_linklist object pointer
i	rotation index count

**Warning**

While exported, it is recommend to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_rotate().

Here is the call graph for this function:



### 38.74.2.19 jit\_linklist\_shuffle()

```
void jit_linklist_shuffle (
    t_jit_linklist * x )
```

Randomizes the order of objects in the linked list.

#### Parameters

x	t_jit_linklist object pointer
---	-------------------------------

#### Warning

While exported, it is recommend to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_shuffle().

Here is the call graph for this function:



### 38.74.2.20 jit\_linklist\_sort()

```
void jit_linklist_sort (
    t_jit_linklist * x,
    long cmpfnvoid *, void * )
```

Sorts linked list based on the provided comparison function.

#### Parameters

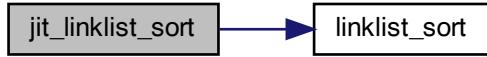
x	t_jit_linklist object pointer
cmpfn	comparison function pointer (returns 0 if a>b, otherwise 1)

#### Warning

While exported, it is recommended to use jit\_object\_method to call methods on an object when the object may not be an instance of t\_jit\_linklist, but instead an object that supports some portion of the t\_jit\_linklist interface. One instance where this is the case is inside of a MOP matrix\_calc method, where the arguments can be either an instance of t\_jit\_linklist, or t\_jit\_matrix which has a getindex method.

References linklist\_sort().

Here is the call graph for this function:



### 38.74.2.21 jit\_linklist\_swap()

```
void jit_linklist_swap (
    t_jit_linklist * x,
    long a,
    long b )
```

Swap list location of the indeces specified.

#### Parameters

x	t_jit_linklist object pointer
a	index a
b	index b

**Warning**

While exported, it is recommended to use `jit_object_method` to call methods on an object when the object may not be an instance of `t_jit_linklist`, but instead an object that supports some portion of the `t_jit_linklist` interface. One instance where this is the case is inside of a MOP `matrix_calc` method, where the arguments can be either an instance of `t_jit_linklist`, or `t_jit_matrix` which has a `getindex` method.

References `linklist_swap()`.

Here is the call graph for this function:



## 38.75 Math Module

Collaboration diagram for Math Module:



### Functions

- double `jit_math_cos` (double x)  
*Calculates the cosine.*
- double `jit_math_sin` (double x)  
*Calculates the sine.*
- double `jit_math_tan` (double x)  
*Calculates the tangent.*
- double `jit_math_acos` (double x)  
*Calculates the arccosine.*
- double `jit_math_asin` (double x)  
*Calculates the arcsine.*
- double `jit_math_atan` (double x)

- double `jit_math_atan2` (double y, double x)  
*Calculates the arctangent.*
- double `jit_math_cosh` (double x)  
*Calculates the four quadrant arctangent.*
- double `jit_math_sinh` (double x)  
*Calculates the hyperbolic cosine.*
- double `jit_math_tanh` (double x)  
*Calculates the hyperbolic sine.*
- double `jit_math_acosh` (double x)  
*Calculates the hyperbolic tangent.*
- double `jit_math_asinh` (double x)  
*Calculates the hyperbolic arccosine.*
- double `jit_math_atanh` (double x)  
*Calculates the hyperbolic arcsine.*
- double `jit_math_exp` (double x)  
*Calculates the exponent.*
- double `jit_math_expm1` (double x)  
*Calculates the exponent minus 1.*
- double `jit_math_exp2` (double x)  
*Calculates the exponent base 2.*
- double `jit_math_log` (double x)  
*Calculates the logarithm.*
- double `jit_math_log2` (double x)  
*Calculates the logarithm base 2.*
- double `jit_math_log10` (double x)  
*Calculates the logarithm base 10.*
- double `jit_math_hypot` (double x, double y)  
*Calculates the hypotenuse.*
- double `jit_math_pow` (double x, double y)  
*Calculates x raised to the y power.*
- double `jit_math_sqrt` (double x)  
*Calculates the square root.*
- double `jit_math_ceil` (double x)  
*Calculates the ceiling.*
- double `jit_math_floor` (double x)  
*Calculates the floor.*
- double `jit_math_round` (double x)  
*Rounds the input.*
- double `jit_math_trunc` (double x)  
*Truncates the input.*
- double `jit_math_fmod` (double x, double y)  
*Calculates the floating point x modulo y.*
- double `jit_math_fold` (double x, double lo, double hi)  
*Calculates the fold of x between lo and hi.*
- double `jit_math_wrap` (double x, double lo, double hi)  
*Calculates the wrap of x between lo and hi.*

- double [jit\\_math\\_j1\\_0](#) (double x)  
*Calculates the j1\_0 Bessel function.*
- double [jit\\_math\\_p1](#) (double x)  
*Calculates the p1 Bessel function.*
- double [jit\\_math\\_q1](#) (double x)  
*Calculates the q1 Bessel function.*
- double [jit\\_math\\_j1](#) (double x)  
*Calculates the j1 Bessel function.*
- unsigned long [jit\\_math\\_roundup\\_poweroftwo](#) (unsigned long x)  
*Rounds up to the nearest power of two.*
- long [jit\\_math\\_is\\_finite](#) (float v)  
*Checks if input is finite.*
- long [jit\\_math\\_is\\_nan](#) (float v)  
*Checks if input is not a number (NaN).*
- long [jit\\_math\\_is\\_valid](#) (float v)  
*Checks if input is both finite and a number.*
- long [jit\\_math\\_is\\_poweroftwo](#) (long x)  
*Checks if input is a power of two.*
- float [jit\\_math\\_fast\\_sqrt](#) (float n)  
*Calculates the square root by fast approximation.*
- float [jit\\_math\\_fast\\_invsqrt](#) (float x)  
*Calculates the inverse square root by fast approximation.*
- float [jit\\_math\\_fast\\_sin](#) (float x)  
*Calculates the sine by fast approximation.*
- float [jit\\_math\\_fast\\_cos](#) (float x)  
*Calculates the cosine by fast approximation.*
- float [jit\\_math\\_fast\\_tan](#) (float x)  
*Calculates the tangent by fast approximation.*
- float [jit\\_math\\_fast\\_asin](#) (float x)  
*Calculates the arcsine by fast approximation.*
- float [jit\\_math\\_fast\\_acos](#) (float x)  
*Calculates the arccosine by fast approximation.*
- float [jit\\_math\\_fast\\_atan](#) (float x)  
*Calculates the arctangent by fast approximation.*

### 38.75.1 Detailed Description

### 38.75.2 Function Documentation

#### 38.75.2.1 [jit\\_math\\_acos\(\)](#)

```
double jit_math_acos (
    double x )
```

Calculates the arccosine.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.2 jit\_math\_acosh()**

```
double jit_math_acosh (
    double x )
```

Calculates the hyperbolic arccosine.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.3 jit\_math\_asin()**

```
double jit_math_asin (
    double x )
```

Calculates the arcsine.

**Parameters**

x	input
---	-------

**Returns**

output

### 38.75.2.4 jit\_math\_asinh()

```
double jit_math_asinh (
    double x )
```

Calculates the hyperbolic arcsine.

#### Parameters

x	input
---	-------

#### Returns

output

### 38.75.2.5 jit\_math\_atan()

```
double jit_math_atan (
    double x )
```

Calculates the arctangent.

#### Parameters

x	input
---	-------

#### Returns

output

### 38.75.2.6 jit\_math\_atan2()

```
double jit_math_atan2 (
    double y,
    double x )
```

Calculates the four quadrant arctangent.

#### Parameters

y	input
x	input

Returns

output

### 38.75.2.7 jit\_math\_atanh()

```
double jit_math_atanh (
    double x )
```

Calculates the hyperbolic arctangent.

Parameters

x	input
---	-------

Returns

output

### 38.75.2.8 jit\_math\_ceil()

```
double jit_math_ceil (
    double x )
```

Calculates the ceiling.

Parameters

x	input
---	-------

Returns

output

### 38.75.2.9 jit\_math\_cos()

```
double jit_math_cos (
    double x )
```

Calculates the cosine.

**Parameters**

x	input
---	-------

**Returns**

output

Referenced by jit\_math\_j1().

**38.75.2.10 jit\_math\_cosh()**

```
double jit_math_cosh (
    double x )
```

Calculates the hyperbolic cosine.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.11 jit\_math\_exp()**

```
double jit_math_exp (
    double x )
```

Calculates the exponent.

**Parameters**

x	input
---	-------

**Returns**

output

### 38.75.2.12 jit\_math\_exp2()

```
double jit_math_exp2 (
    double x )
```

Calculates the exponent base 2.

#### Parameters

x	input
---	-------

#### Returns

output

### 38.75.2.13 jit\_math\_expm1()

```
double jit_math_expm1 (
    double x )
```

Calculates the exponent minus 1.

#### Parameters

x	input
---	-------

#### Returns

output

### 38.75.2.14 jit\_math\_fast\_acos()

```
float jit_math_fast_acos (
    float x )
```

Calculates the arccosine by fast approximation.

Absolute error of 6.8e-05 for [0, 1]

**Parameters**

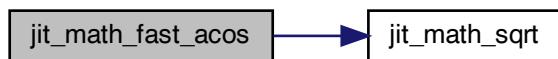
x	input
---	-------

**Returns**

output

References jit\_math\_sqrt().

Here is the call graph for this function:

**38.75.2.15 jit\_math\_fast\_asin()**

```
float jit_math_fast_asin (
    float x )
```

Calculates the arcsine by fast approximation.

Absolute error of 6.8e-05 for [0, 1]

**Parameters**

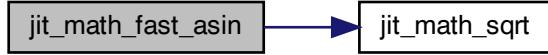
x	input
---	-------

**Returns**

output

References jit\_math\_sqrt().

Here is the call graph for this function:



### 38.75.2.16 jit\_math\_fast\_atan()

```
float jit_math_fast_atan (
    float x )
```

Calculates the arctangent by fast approximation.

Absolute error of 1.43-08 for [-1, 1]

#### Parameters

x	input
---	-------

#### Returns

output

### 38.75.2.17 jit\_math\_fast\_cos()

```
float jit_math_fast_cos (
    float x )
```

Calculates the cosine by fast approximation.

Absolute error of 1.2e-03 for [0, PI/2]

#### Parameters

x	input
---	-------

**Returns**

output

**38.75.2.18 jit\_math\_fast\_invsqrt()**

```
float jit_math_fast_invsqrt (
    float x )
```

Calculates the inverse square root by fast approximation.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.19 jit\_math\_fast\_sin()**

```
float jit_math_fast_sin (
    float x )
```

Calculates the sine by fast approximation.

Absolute error of 1.7e-04 for [0, PI/2]

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.20 jit\_math\_fast\_sqrt()**

```
float jit_math_fast_sqrt (
    float n )
```

Calculates the square root by fast approximation.

**Parameters**

<code>n</code>	input
----------------	-------

**Returns**

output

**38.75.2.21 jit\_math\_fast\_tan()**

```
float jit_math_fast_tan (
    float x )
```

Calculates the tangent by fast approximation.

Absolute error of 1.9e-00 for [0, PI/4]

**Parameters**

<code>x</code>	input
----------------	-------

**Returns**

output

**38.75.2.22 jit\_math\_floor()**

```
double jit_math_floor (
    double x )
```

Calculates the floor.

**Parameters**

<code>x</code>	input
----------------	-------

**Returns**

output

**38.75.2.23 jit\_math\_fmod()**

```
double jit_math_fmod (
    double x,
    double y )
```

Calculates the floating point x modulo y.

**Parameters**

x	input
y	input

**Returns**

output

**38.75.2.24 jit\_math\_fold()**

```
double jit_math_fold (
    double x,
    double lo,
    double hi )
```

Calculates the fold of x between lo and hi.

**Parameters**

x	input
lo	lower bound
hi	upper bound

**Returns**

output

**38.75.2.25 jit\_math\_hypot()**

```
double jit_math_hypot (
    double x,
    double y )
```

Calculates the hypotenuse.

**Parameters**

x	input
y	input

**Returns**

output

**38.75.2.26 jit\_math\_is\_finite()**

```
long jit_math_is_finite (
    float v )
```

Checks if input is finite.

**Parameters**

v	input
---	-------

**Returns**

1 if finite. Otherwise, 0.

Referenced by jit\_math\_is\_valid().

**38.75.2.27 jit\_math\_is\_nan()**

```
long jit_math_is_nan (
    float v )
```

Checks if input is not a number (NaN).

**Parameters**

v	input
---	-------

**Returns**

1 if not a number. Otherwise, 0.

Referenced by jit\_math\_is\_valid().

### 38.75.2.28 jit\_math\_is\_poweroftwo()

```
long jit_math_is_poweroftwo (
    long x )
```

Checks if input is a power of two.

#### Parameters

x	input
---	-------

#### Returns

1 if finite. Otherwise, 0.

### 38.75.2.29 jit\_math\_is\_valid()

```
long jit_math_is_valid (
    float v )
```

Checks if input is both finite and a number.

#### Parameters

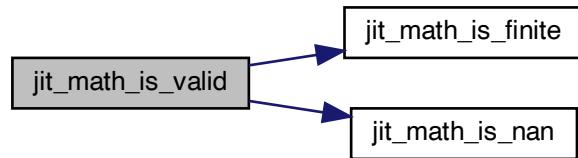
v	input
---	-------

**Returns**

1 if valid. Otherwise, 0.

References jit\_math\_is\_finite(), and jit\_math\_is\_nan().

Here is the call graph for this function:



### 38.75.2.30 jit\_math\_j1()

```
double jit_math_j1 ( double x )
```

Calculates the j1 Bessel function.

**Parameters**

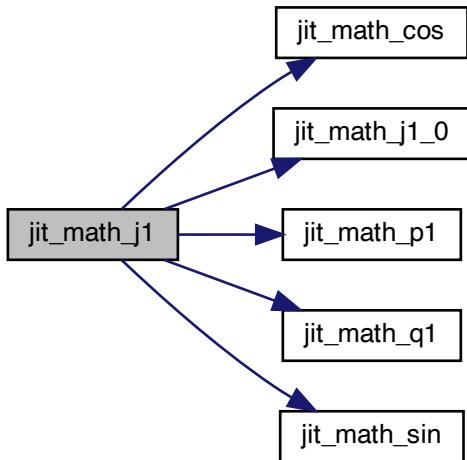
x	input
---	-------

**Returns**

output

References jit\_math\_cos(), jit\_math\_j1\_0(), jit\_math\_p1(), jit\_math\_q1(), and jit\_math\_sin().

Here is the call graph for this function:



### 38.75.2.31 jit\_math\_j1\_0()

```
double jit_math_j1_0 (
    double x )
```

Calculates the j1\_0 Bessel function.

**Parameters**

x	input
---	-------

**Returns**

output

Referenced by jit\_math\_j1().

**38.75.2.32 jit\_math\_log()**

```
double jit_math_log (
    double x )
```

Calculates the logarithm.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.33 jit\_math\_log10()**

```
double jit_math_log10 (
    double x )
```

Calculates the logarithm base 10.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.34 jit\_math\_log2()**

```
double jit_math_log2 (
    double x )
```

Calculates the logarithm base 2.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.35 jit\_math\_p1()**

```
double jit_math_p1 (
    double x )
```

Calcuates the p1 Bessel function.

**Parameters**

x	input
---	-------

**Returns**

output

Referenced by jit\_math\_j1().

**38.75.2.36 jit\_math\_pow()**

```
double jit_math_pow (
    double x,
    double y )
```

Calculates x raised to the y power.

**Parameters**

x	input
y	input

**Returns**

output

**38.75.2.37 jit\_math\_q1()**

```
double jit_math_q1 (
    double x )
```

Calculates the q1 Bessel function.

**Parameters**

x	input
---	-------

**Returns**

output

Referenced by jit\_math\_j1().

**38.75.2.38 jit\_math\_round()**

```
double jit_math_round (
    double x )
```

Rounds the input.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.39 jit\_math\_roundup\_poweroftwo()**

```
unsigned long jit_math_roundup_poweroftwo (
    unsigned long x )
```

Rounds up to the nearest power of two.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.40 jit\_math\_sin()**

```
double jit_math_sin (
    double x )
```

Calculates the sine.

**Parameters**

x	input
---	-------

**Returns**

output

Referenced by jit\_math\_j1().

**38.75.2.41 jit\_math\_sinh()**

```
double jit_math_sinh (
    double x )
```

Calculates the hyperbolic sine.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.42 jit\_math\_sqrt()**

```
double jit_math_sqrt (
    double x )
```

Calculates the square root.

**Parameters**

x	input
---	-------

**Returns**

output

Referenced by jit\_math\_fast\_acos(), and jit\_math\_fast\_asin().

**38.75.2.43 jit\_math\_tan()**

```
double jit_math_tan (
    double x )
```

Calculates the tangent.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.44 jit\_math\_tanh()**

```
double jit_math_tanh (
    double x )
```

Calculates the hyperbolic tangent.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.45 jit\_math\_trunc()**

```
double jit_math_trunc (
    double x )
```

Truncates the input.

**Parameters**

x	input
---	-------

**Returns**

output

**38.75.2.46 jit\_math\_wrap()**

```
double jit_math_wrap (
    double x,
    double lo,
    double hi )
```

Calculates the wrap of x between lo and hi.

**Parameters**

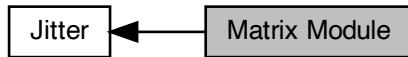
x	input
lo	lower bound
hi	upper bound

Returns

output

## 38.76 Matrix Module

Collaboration diagram for Matrix Module:



### Functions

- void [jit\\_linklist\\_free \(t\\_jit\\_linklist \\*x\)](#)  
*Frees instance of t\_jit\_linklist.*
- void \* [jit\\_matrix\\_new \(t\\_jit\\_matrix\\_info \\*info\)](#)  
*Constructs instance of t\_jit\_matrix.*
- void \* [jit\\_matrix\\_newcopy \(t\\_jit\\_matrix \\*copyme\)](#)  
*Constructs instance of t\_jit\_matrix, copying from input.*
- t\_jit\_err [jit\\_matrix\\_free \(t\\_jit\\_matrix \\*x\)](#)  
*Frees instance of t\_jit\_matrix.*
- t\_jit\_err [jit\\_matrix\\_setinfo \(t\\_jit\\_matrix \\*x, t\\_jit\\_matrix\\_info \\*info\)](#)  
*Sets all attributes according to the t\_jit\_matrix\_info struct provided.*
- t\_jit\_err [jit\\_matrix\\_setinfo\\_ex \(t\\_jit\\_matrix \\*x, t\\_jit\\_matrix\\_info \\*info\)](#)  
*Sets all attributes according to the t\_jit\_matrix\_info struct provided (including data flags).*
- t\_jit\_err [jit\\_matrix\\_getinfo \(t\\_jit\\_matrix \\*x, t\\_jit\\_matrix\\_info \\*info\)](#)  
*Retrieves all attributes, copying into the t\_jit\_matrix\_info struct provided.*
- t\_jit\_err [jit\\_matrix\\_getdata \(t\\_jit\\_matrix \\*x, void \\*\\*data\)](#)  
*Retrieves matrix data pointer.*
- t\_jit\_err [jit\\_matrix\\_data \(t\\_jit\\_matrix \\*x, void \\*data\)](#)  
*Sets matrix data pointer.*
- t\_jit\_err [jit\\_matrix\\_freedata \(t\\_jit\\_matrix \\*x\)](#)  
*Frees matrix's internal data pointer if an internal reference and sets to NULL.*
- t\_jit\_err [jit\\_matrix\\_info\\_default \(t\\_jit\\_matrix\\_info \\*info\)](#)  
*Initializes matrix info struct to default values.*
- t\_jit\_err [jit\\_matrix\\_clear \(t\\_jit\\_matrix \\*x\)](#)  
*Sets all cells in matrix to the zero.*
- t\_jit\_err [jit\\_matrix\\_setcell1d \(t\\_jit\\_matrix \\*x, t\\_symbol \\*s, long argc, t\\_atom \\*argv\)](#)  
*Sets cell at index to the value provided.*
- t\_jit\_err [jit\\_matrix\\_setcell2d \(t\\_jit\\_matrix \\*x, t\\_symbol \\*s, long argc, t\\_atom \\*argv\)](#)

- `t_jit_err jit_matrix_setcell3d (t_jit_matrix *x, t_symbol *s, long argc, t_atom *argv)`

*Sets cell at index to the value provided.*
- `t_jit_err jit_matrix_setplane1d (t_jit_matrix *x, t_symbol *s, long argc, t_atom *argv)`

*Sets plane of cell at index to the value provided.*
- `t_jit_err jit_matrix_setplane2d (t_jit_matrix *x, t_symbol *s, long argc, t_atom *argv)`

*Sets plane of cell at index to the value provided.*
- `t_jit_err jit_matrix_setplane3d (t_jit_matrix *x, t_symbol *s, long argc, t_atom *argv)`

*Sets plane of cell at index to the value provided.*
- `t_jit_err jit_matrix_setcell (t_jit_matrix *x, t_symbol *s, long argc, t_atom *argv)`

*Sets cell at index to the value provided.*
- `t_jit_err jit_matrix_getcell (t_jit_matrix *x, t_symbol *s, long argc, t_atom *argv, long *rac, t_atom **rav)`

*Gets cell at index to the value provided.*
- `t_jit_err jit_matrix_setall (t_jit_matrix *x, t_symbol *s, long argc, t_atom *argv)`

*Sets all cells to the value provided.*
- `t_jit_err jit_matrix_fillplane (t_jit_matrix *x, t_symbol *s, long argc, t_atom *argv)`

*Sets the plane specified in all cells to the value provided.*
- `t_jit_err jit_matrix_frommatrix (t_jit_matrix *dst_matrix, t_jit_matrix *src_matrix, t_matrix_conv_info *mcinfo)`

*Copies Jitter matrix data from another matrix.*
- `t_jit_err jit_matrix_op (t_jit_matrix *x, t_symbol *s, long argc, t_atom *argv)`

*Applies unary or binary operator to matrix See Jitter user documentation for more information.*
- `t_jit_err jit_matrix_exprfill (t_jit_matrix *x, t_symbol *s, long argc, t_atom *argv)`

*Fills cells according to the jit.expr expression provided.*
- `t_jit_err jit_matrix_jit_gl_texture (t_jit_matrix *x, t_symbol *s, long argc, t_atom *argv)`

*Copies texture information to matrix.*

### 38.76.1 Detailed Description

### 38.76.2 Function Documentation

#### 38.76.2.1 jit\_linklist\_free()

```
void jit_linklist_free (
    t_jit_linklist * x )
```

Frees instance of `t_jit_linklist`.

should never be called directly

#### Parameters

x	<code>t_jit_linklist</code> object pointer
---	--

**Returns**

t\_jit\_err error code

**Warning**

Use jit\_object\_free instead.

References linklist\_clear().

Here is the call graph for this function:



### 38.76.2.2 jit\_matrix\_clear()

```
t_jit_err jit_matrix_clear (
    t_jit_matrix * x )
```

Sets all cells in matrix to the zero.

See Jitter user documentation for more information.

**Parameters**

x	t_jit_matrix object pointer
---	-----------------------------

**Returns**

t\_jit\_err error code

**Warning**

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of t\_jit\_matrix.

### 38.76.2.3 jit\_matrix\_data()

```
t_jit_err jit_matrix_data (
    t_jit_matrix * x,
    void * data )
```

Sets matrix data pointer.

#### Parameters

<i>x</i>	<code>t_jit_matrix</code> object pointer
<i>data</i>	data pointer

#### Returns

`t_jit_err` error code

#### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

### 38.76.2.4 jit\_matrix\_exprfill()

```
t_jit_err jit_matrix_exprfill (
    t_jit_matrix * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Fills cells according to the `jit.expr` expression provided.

See Jitter user documentation for more information.

#### Parameters

<i>x</i>	<code>t_jit_matrix</code> object pointer
<i>s</i>	message symbol pointer
<i>argc</i>	argument count
<i>argv</i>	argument vector

#### Returns

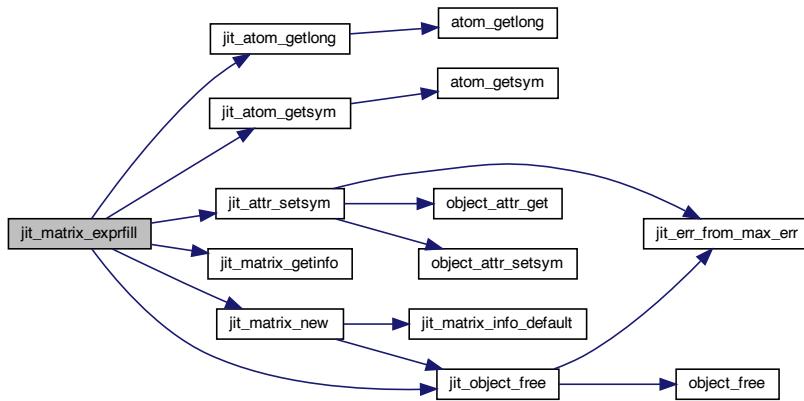
`t_jit_err` error code

### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

References `_jit_sym_float32`, `_jit_sym_frommatrix`, `_jit_sym_matrix_calc`, `_jit_sym_nothing`, `A_SYM`, `t_matrix` ↔ `conv_info::flags`, `jit_atom_getlong()`, `jit_atom_getsym()`, `jit_attr_setsym()`, `jit_matrix_getinfo()`, `JIT_MATRIX_MAX` ↔ `PLANECOUNT`, `jit_matrix_new()`, `jit_object_free()`, `t_jit_matrix_info::planecount`, `t_matrix_conv_info::planemap`, and `t_jit_matrix_info::type`.

Here is the call graph for this function:



### 38.76.2.5 jit\_matrix\_fillplane()

```
t_jit_err jit_matrix_fillplane (
    t_jit_matrix * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Sets the plane specified in all cells to the value provided.

See Jitter user documentation for more information.

#### Parameters

<code>x</code>	<code>t_jit_matrix</code> object pointer
<code>s</code>	message symbol pointer
<code>argc</code>	argument count
<code>argv</code>	argument vector

**Returns**

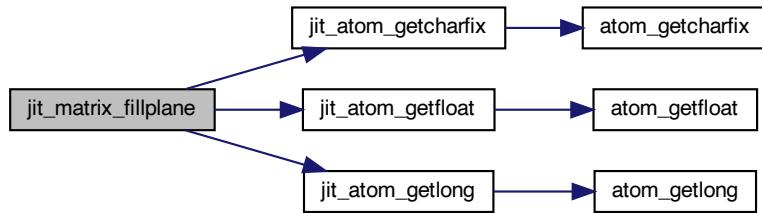
`t_jit_err` error code

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

References `_jit_sym_char`, `_jit_sym_float32`, `_jit_sym_float64`, `_jit_sym_long`, `jit_atom_getcharfix()`, `jit_atom_getfloat()`, and `jit_atom_getlong()`.

Here is the call graph for this function:

**38.76.2.6 `jit_matrix_free()`**

```
t_jit_err jit_matrix_free (
    t_jit_matrix * x )
```

Frees instance of `t_jit_matrix`.

**Parameters**

<code>x</code>	<code>t_jit_matrix</code> object pointer
----------------	--

**Returns**

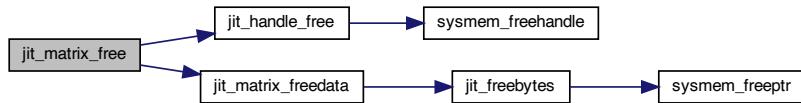
`t_jit_err` error code

### Warning

Use `jit_object_free` instead.

References `jit_handle_free()`, `JIT_MATRIX_DATA_HANDLE`, `JIT_MATRIX_DATA_REFERENCE`, and `jit_matrix_free()`.

Here is the call graph for this function:



### 38.76.2.7 `jit_matrix_freedata()`

```
t_jit_err jit_matrix_freedata (
    t_jit_matrix * x )
```

Frees matrix's internal data pointer if an internal reference and sets to NULL.

#### Parameters

<code>x</code>	<code>t_jit_matrix</code> object pointer
----------------	--

#### Returns

`t_jit_err` error code

### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

References `jit_freebytes()`, and `JIT_MATRIX_DATA_REFERENCE`.

Referenced by `jit_matrix_free()`.

Here is the call graph for this function:



### 38.76.2.8 jit\_matrix\_frommatrix()

```
t_jit_err jit_matrix_frommatrix (
    t_jit_matrix * dst_matrix,
    t_jit_matrix * src_matrix,
    t_matrix_conv_info * mcinfo )
```

Copies Jitter matrix data from another matrix.

#### Parameters

<i>dst_matrix</i>	destination t_jit_matrix object pointer
<i>src_matrix</i>	destination t_jit_matrix object pointer
<i>mcinfo</i>	conversion information pointer

#### Returns

t\_jit\_err error code

#### Warning

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of t\_jit\_matrix.

References `_jit_sym_getdata`, `_jit_sym_getinfo`, `_jit_sym_lock`, `t_jit_matrix_info::dim`, `t_jit_matrix_info::dimcount`, `t_matrix_conv_info::dstdimend`, `t_matrix_conv_info::dstdimstart`, `t_matrix_conv_info::flags`, `JIT_MATRIX_CONVERT_DSTDIM`, `JIT_MATRIX_CONVERT_SRCDIM`, `JIT_MATRIX_MAX_DIMCOUNT`, `MAX`, `t_jit_matrix_info::size`, `t_matrix_conv_info::srcdimend`, and `t_matrix_conv_info::srcdimstart`.

### 38.76.2.9 jit\_matrix\_getcell()

```
t_jit_err jit_matrix_getcell (
    t_jit_matrix * x,
    t_symbol * s,
    long argc,
    t_atom * argv,
    long * rac,
    t_atom ** rav )
```

Gets cell at index to the value provided.

See Jitter user documentation for more information.

#### Parameters

<i>x</i>	t_jit_matrix object pointer
<i>s</i>	message symbol pointer
<i>argc</i>	argument count
<i>argv</i>	argument vector
<i>rac</i>	return value atom count
<i>rav</i>	return value atom vector

#### Returns

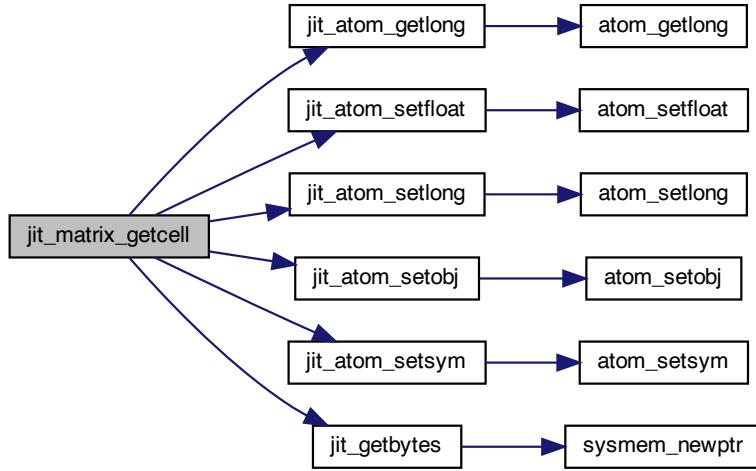
t\_jit\_err error code

#### Warning

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of t\_jit\_matrix.

References \_jit\_sym\_char, \_jit\_sym\_float32, \_jit\_sym\_float64, \_jit\_sym\_long, \_jit\_sym\_object, \_jit\_sym\_symbol, \_jit\_sym\_val, jit\_atom\_getlong(), jit\_atom\_setfloat(), jit\_atom\_setlong(), jit\_atom\_setobj(), jit\_atom\_setsym(), jit\_getbytes(), and JIT\_MATRIX\_MAX\_DIMCOUNT.

Here is the call graph for this function:



### 38.76.2.10 `jit_matrix_getdata()`

```
t_jit_err jit_matrix_getdata (
    t_jit_matrix * x,
    void ** data )
```

Retrieves matrix data pointer.

#### Parameters

<code>x</code>	<code>t_jit_matrix</code> object pointer
<code>data</code>	pointer to data pointer (set to NULL if matrix is not available)

#### Returns

`t_jit_err` error code

#### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

### 38.76.2.11 jit\_matrix\_getinfo()

```
t_jit_err jit_matrix_getinfo (
    t_jit_matrix * x,
    t_jit_matrix_info * info )
```

Retrieves all attributes, copying into the [t\\_jit\\_matrix\\_info](#) struct provided.

#### Parameters

x	t_jit_matrix object pointer
info	<a href="#">t_jit_matrix_info</a> pointer

#### Returns

t\_jit\_err error code

#### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

Referenced by `jit_matrix_exprfill()`, and `jit_matrix_op()`.

### 38.76.2.12 jit\_matrix\_info\_default()

```
t_jit_err jit_matrix_info_default (
    t_jit_matrix_info * info )
```

Initializes matrix info struct to default values.

#### Parameters

info	<a href="#">t_jit_matrix_info</a> struct pointer
------	--

#### Returns

t\_jit\_err error code

#### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

References `t_jit_matrix_info::dim`, `t_jit_matrix_info::dimcount`, `t_jit_matrix_info::dimstride`, `t_jit_matrix_info::flags`, `JIT_MATRIX_MAX_DIMCOUNT`, `t_jit_matrix_info::planecount`, `t_jit_matrix_info::size`, and `t_jit_matrix_info::type`.

Referenced by `jit_matrix_new()`, `max_jit_mop_inputs()`, `max_jit_mop_matrix_args()`, and `max_jit_mop_outputs()`.

### 38.76.2.13 `jit_matrix_jit_gl_texture()`

```
t_jit_err jit_matrix_jit_gl_texture (
    t_jit_matrix * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Copies texture information to matrix.

See Jitter user documentation for more information.

#### Parameters

<code>x</code>	<code>t_jit_matrix</code> object pointer
<code>s</code>	message symbol pointer
<code>argc</code>	argument count
<code>argv</code>	argument vector

#### Returns

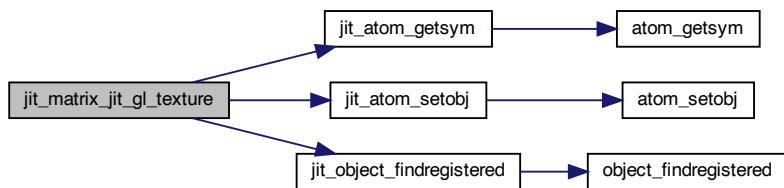
`t_jit_err` error code

#### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

References `_jit_sym_nothing`, `jit_atom_getsym()`, `jit_atom_setobj()`, and `jit_object_findregistered()`.

Here is the call graph for this function:



### 38.76.2.14 jit\_matrix\_new()

```
void * jit_matrix_new (
    t_jit_matrix_info * info )
```

Constructs instance of t\_jit\_matrix.

#### Parameters

<i>info</i>	t_jit_matrix_info struct pointer
-------------	----------------------------------

#### Returns

t\_jit\_matrix object pointer

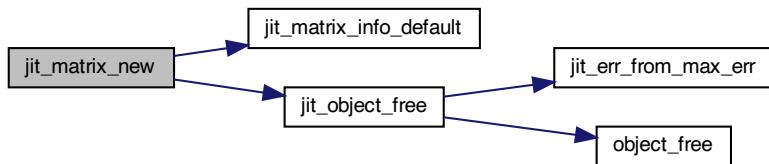
#### Warning

This function is not exported, but is provided for reference when calling via jit\_object\_new.

References t\_jit\_matrix\_info::flags, JIT\_MATRIX\_DATA\_FLAGS\_USE, JIT\_MATRIX\_DATA\_HANDLE, JIT\_MATRIX\_DATA\_REFERENCE, jit\_matrix\_info\_default(), and jit\_object\_free().

Referenced by jit\_matrix\_exprfill(), and jit\_matrix\_op().

Here is the call graph for this function:



### 38.76.2.15 jit\_matrix\_newcopy()

```
void * jit_matrix_newcopy (
    t_jit_matrix * copyme )
```

Constructs instance of t\_jit\_matrix, copying from input.

**Parameters**

<i>copyme</i>	t_jit_matrix object pointer
---------------	-----------------------------

**Returns**

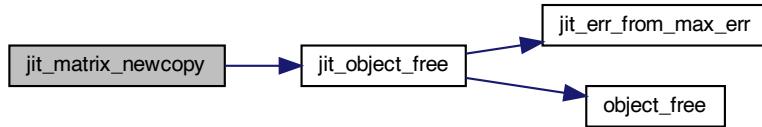
t\_jit\_matrix object pointer

**Warning**

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of t\_jit\_matrix.

References JIT\_MATRIX\_DATA\_REFERENCE, and jit\_object\_free().

Here is the call graph for this function:

**38.76.2.16 jit\_matrix\_op()**

```
t_jit_err jit_matrix_op (
    t_jit_matrix * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Applies unary or binary operator to matrix See Jitter user documentation for more information.

**Parameters**

<i>x</i>	t_jit_matrix object pointer
<i>s</i>	message symbol pointer
<i>argc</i>	argument count
<i>argv</i>	argument vector

**Returns**

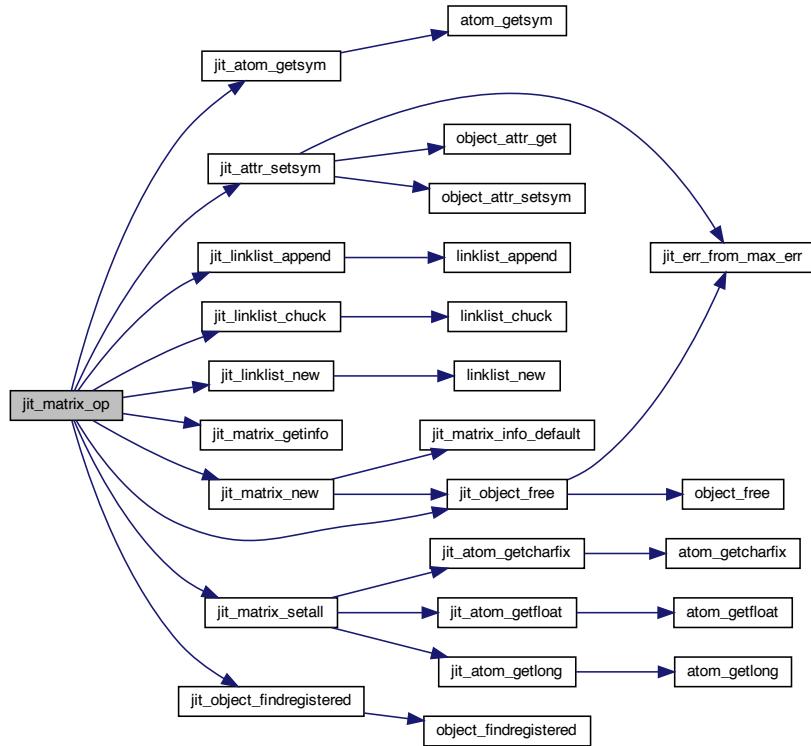
`t_jit_err` error code

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

References `_jit_sym_class_jit_matrix`, `_jit_sym_matrix_calc`, `_jit_sym_nothing`, `A_FLOAT`, `A_LONG`, `A_OBJ`, `A_SYM`, `jit_atom_getsym()`, `jit_attr_setsym()`, `jit_linklist_append()`, `jit_linklist_chuck()`, `jit_linklist_new()`, `jit_matrix_getinfo()`, `jit_matrix_new()`, `jit_matrix_setall()`, `jit_object_findregistered()`, `jit_object_free()`, and `t_symbol::s_name`.

Here is the call graph for this function:

**38.76.2.17 jit\_matrix\_setall()**

```
t_jit_err jit_matrix_setall (
    t_jit_matrix * x,
```

```
t_symbol * s,  
long argc,  
t_atom * argv )
```

Sets all cells to the value provided.

See Jitter user documentation for more information.

**Parameters**

<i>x</i>	t_jit_matrix object pointer
<i>s</i>	message symbol pointer
<i>argc</i>	argument count
<i>argv</i>	argument vector

**Returns**

t\_jit\_err error code

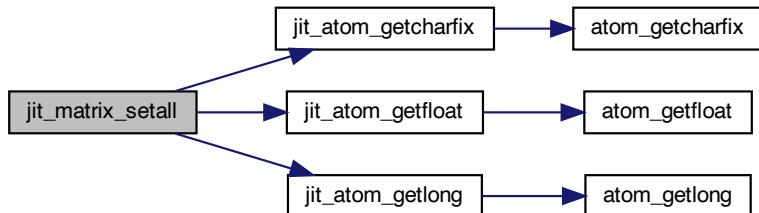
**Warning**

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of t\_jit\_matrix.

References \_jit\_sym\_char, \_jit\_sym\_float32, \_jit\_sym\_float64, \_jit\_sym\_long, jit\_atom\_getcharfix(), jit\_atom\_getfloat(), jit\_atom\_getlong(), and JIT\_MATRIX\_MAX\_PLANECOUNT.

Referenced by jit\_matrix\_op().

Here is the call graph for this function:

**38.76.2.18 jit\_matrix\_setcell()**

```
t_jit_err jit_matrix_setcell (
    t_jit_matrix * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Sets cell at index to the value provided.

See Jitter user documentation for more information.

**Parameters**

<i>x</i>	t_jit_matrix object pointer
<i>s</i>	message symbol pointer
<i>argc</i>	argument count
<i>argv</i>	argument vector

**Returns**

t\_jit\_err error code

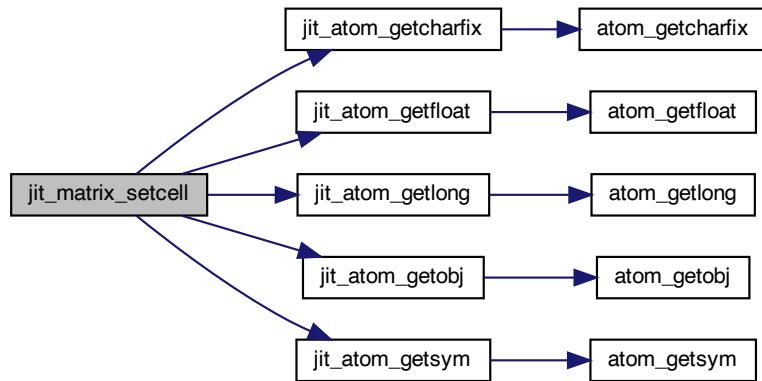
**Warning**

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of t\_jit\_matrix.

References \_jit\_sym\_char, \_jit\_sym\_float32, \_jit\_sym\_float64, \_jit\_sym\_long, \_jit\_sym\_object, \_jit\_sym\_plane, \_jit\_sym\_symbol, \_jit\_sym\_val, A\_SYM, jit\_atom\_getcharfix(), jit\_atom\_getfloat(), jit\_atom\_getlong(), jit\_atom\_getobj(), jit\_atom\_getsym(), JIT\_MATRIX\_MAX\_DIMCOUNT, and word::w\_sym.

Referenced by jit\_matrix\_setcell1d(), jit\_matrix\_setcell2d(), jit\_matrix\_setcell3d(), jit\_matrix\_setplane1d(), jit\_matrix\_setplane2d(), and jit\_matrix\_setplane3d().

Here is the call graph for this function:



### 38.76.2.19 jit\_matrix\_setcell1d()

```
t_jit_err jit_matrix_setcell1d (
    t_jit_matrix * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Sets cell at index to the value provided.

See Jitter user documentation for more information.

#### Parameters

<i>x</i>	t_jit_matrix object pointer
<i>s</i>	message symbol pointer
<i>argc</i>	argument count
<i>argv</i>	argument vector

#### Returns

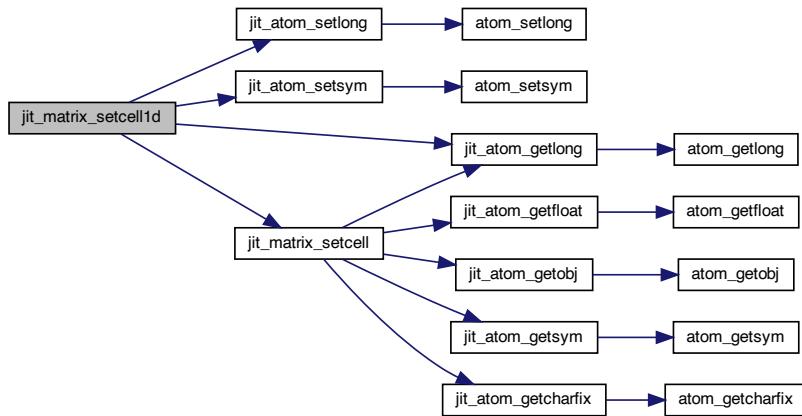
t\_jit\_err error code

#### Warning

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of t\_jit\_matrix.

References jit\_sym\_val, jit\_atom\_getlong(), jit\_atom\_setlong(), jit\_atom\_setsym(), and jit\_matrix\_setcell().

Here is the call graph for this function:



### 38.76.2.20 jit\_matrix\_setcell2d()

```
t_jit_err jit_matrix_setcell2d (
    t_jit_matrix * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Sets cell at index to the value provided.

See Jitter user documentation for more information.

#### Parameters

<i>x</i>	t_jit_matrix object pointer
<i>s</i>	message symbol pointer
<i>argc</i>	argument count
<i>argv</i>	argument vector

#### Returns

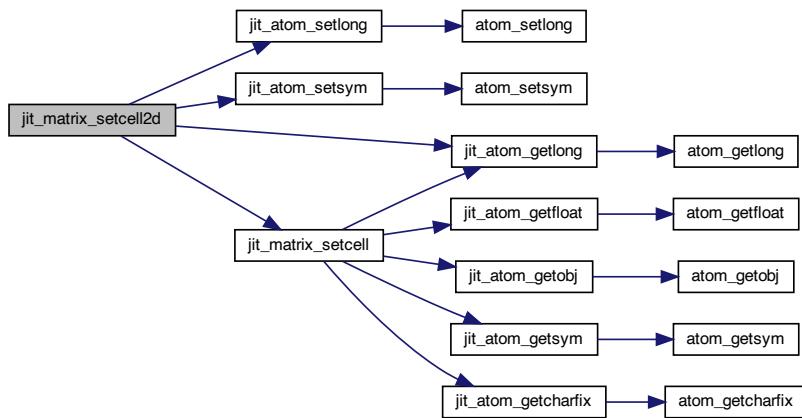
t\_jit\_err error code

#### Warning

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of t\_jit\_matrix.

References jit\_sym\_val, jit\_atom\_getlong(), jit\_atom\_setlong(), jit\_atom\_setsym(), and jit\_matrix\_setcell().

Here is the call graph for this function:



### 38.76.2.21 jit\_matrix\_setcell3d()

```
t_jit_err jit_matrix_setcell3d (
    t_jit_matrix * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Sets cell at index to the value provided.

See Jitter user documentation for more information.

#### Parameters

<i>x</i>	t_jit_matrix object pointer
<i>s</i>	message symbol pointer
<i>argc</i>	argument count
<i>argv</i>	argument vector

#### Returns

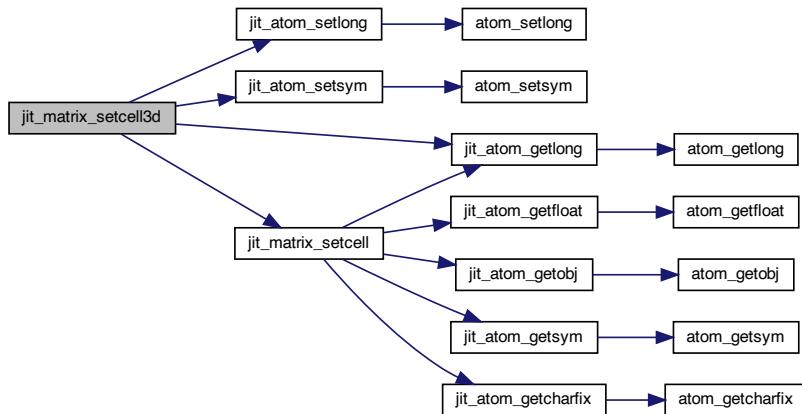
t\_jit\_err error code

#### Warning

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of t\_jit\_matrix.

References jit\_sym\_val, jit\_atom\_getlong(), jit\_atom\_setlong(), jit\_atom\_setsym(), and jit\_matrix\_setcell().

Here is the call graph for this function:



### 38.76.2.22 jit\_matrix\_setinfo()

```
t_jit_err jit_matrix_setinfo (
    t_jit_matrix * x,
    t_jit_matrix_info * info )
```

Sets all attributes according to the [t\\_jit\\_matrix\\_info](#) struct provided.

#### Parameters

<i>x</i>	t_jit_matrix object pointer
<i>info</i>	<a href="#">t_jit_matrix_info</a> pointer

#### Returns

t\_jit\_err error code

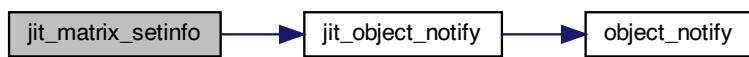
#### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

References `_jit_sym_setinfo`, and `jit_object_notify()`.

Referenced by `jit_matrix_setinfo_ex()`.

Here is the call graph for this function:



### 38.76.2.23 jit\_matrix\_setinfo\_ex()

```
t_jit_err jit_matrix_setinfo_ex (
    t_jit_matrix * x,
    t_jit_matrix_info * info )
```

Sets all attributes according to the [t\\_jit\\_matrix\\_info](#) struct provided (including data flags).

**Parameters**

<i>x</i>	t_jit_matrix object pointer
<i>info</i>	t_jit_matrix_info pointer

**Returns**

t\_jit\_err error code

**Warning**

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of t\_jit\_matrix.

References \_jit\_sym\_modified, \_jit\_sym\_nothing, t\_jit\_matrix\_info::flags, JIT\_MATRIX\_DATA\_REFERENCE, jit\_matrix\_setinfo(), jit\_object\_notify(), and t\_jit\_matrix\_info::size.

Here is the call graph for this function:

**38.76.2.24 jit\_matrix\_setplane1d()**

```
t_jit_err jit_matrix_setplane1d (
    t_jit_matrix * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Sets plane of cell at index to the value provided.

See Jitter user documentation for more information.

**Parameters**

<i>x</i>	t_jit_matrix object pointer
<i>s</i>	message symbol pointer
<i>argc</i>	argument count
<i>argv</i>	argument vector

**Returns**

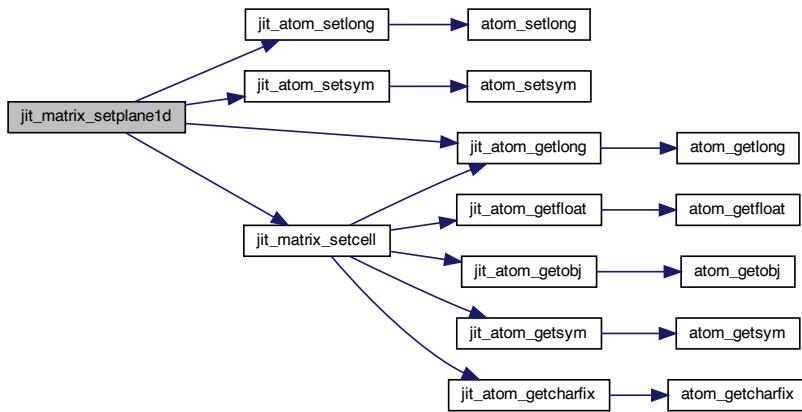
`t_jit_err` error code

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

References `_jit_sym_plane`, `_jit_sym_val`, `jit_atom_getlong()`, `jit_atom_setlong()`, `jit_atom_setsym()`, and `jit_matrix->setcell()`.

Here is the call graph for this function:

**38.76.2.25 jit\_matrix\_setplane2d()**

```
t_jit_err jit_matrix_setplane2d (
    t_jit_matrix * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Sets plane of cell at index to the value provided.

See Jitter user documentation for more information.

**Parameters**

<code>x</code>	<code>t_jit_matrix</code> object pointer
<code>s</code>	message symbol pointer
<code>argc</code>	argument count
<code>argv</code>	argument vector

**Returns**

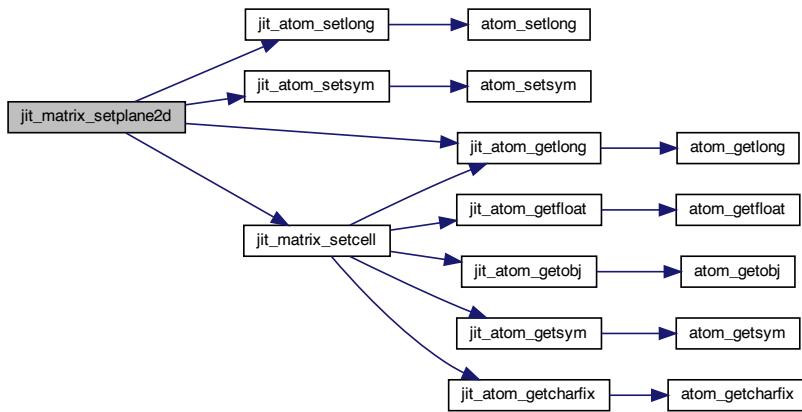
`t_jit_err` error code

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

References `_jit_sym_plane`, `_jit_sym_val`, `jit_atom_getlong()`, `jit_atom_setlong()`, `jit_atom_setsym()`, and `jit_matrix_setcell()`.

Here is the call graph for this function:

**38.76.2.26 jit\_matrix\_setplane3d()**

```
t_jit_err jit_matrix_setplane3d (
    t_jit_matrix * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Sets plane of cell at index to the value provided.

See Jitter user documentation for more information.

**Parameters**

<code>x</code>	<code>t_jit_matrix</code> object pointer
<code>s</code>	message symbol pointer
<code>argc</code>	argument count
<code>argv</code>	argument vector

**Returns**

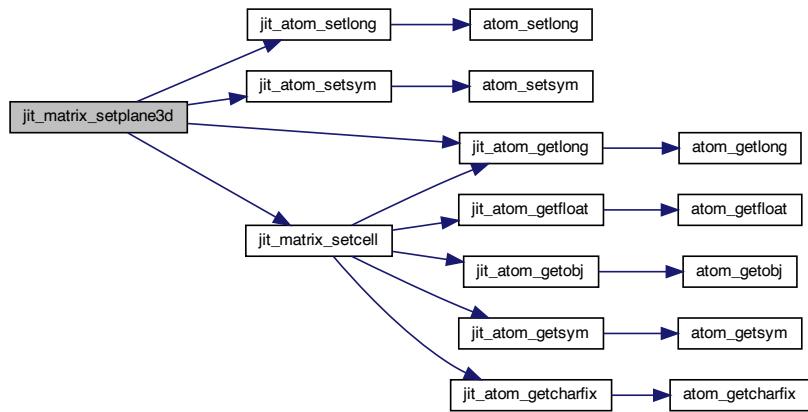
`t_jit_err` error code

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_matrix`.

References `_jit_sym_plane`, `_jit_sym_val`, `jit_atom_getlong()`, `jit_atom_setlong()`, `jit_atom_setsym()`, and `jit_matrix_<-setcell()`.

Here is the call graph for this function:



## 38.77 Max Wrapper Module

Collaboration diagram for Max Wrapper Module:



## Functions

- void **max\_jit\_attr\_set** (void \*x, **t\_symbol** \*s, short ac, **t\_atom** \*av)
 

*Sets attribute value.*
- **t\_jit\_err max\_jit\_attr\_get** (void \*x, **t\_symbol** \*s, long \*ac, **t\_atom** \*\*av)
 

*Retrieves attribute value.*
- void **max\_jit\_attr\_getdump** (void \*x, **t\_symbol** \*s, short argc, **t\_atom** \*argv)
 

*Retrieves attribute value and sends out dump outlet.*
- long **max\_jit\_attr\_args\_offset** (short ac, **t\_atom** \*av)
 

*Determines argument offset to first attribute argument.*
- void **max\_jit\_attr\_args** (void \*x, short ac, **t\_atom** \*av)
 

*Processes attribute arguments.*
- void **max\_jit\_classex\_standard\_wrap** (void \*mclass, void \*jclass, long flags)
 

*Adds standard Jitter methods, as well as public methods and attributes of the specified Jitter class.*
- void **max\_addmethod\_defer** (method m, char \*s)
 

*Adds method to Max class that calls defer rather than the method directly.*
- void **max\_addmethod\_defer\_low** (method m, char \*s)
 

*Adds method to Max class that calls defer\_low rather than the method directly.*
- void **max\_addmethod\_usurp** (method m, char \*s)
 

*Adds method to Max class that uses the usurp mechanism to execute method at low priority without backlog.*
- void **max\_addmethod\_usurp\_low** (method m, char \*s)
 

*Adds method to Max class that uses the usurp mechanism to execute method at low priority without backlog.*
- void \* **max\_jit\_classex\_setup** (long oboffset)
 

*Allocates and initializes special t\_max\_jit\_classex data, used by the Max wrapper class.*
- **t\_jit\_err max\_jit\_classex\_addattr** (void \*x, void \*attr)
 

*Adds an attribute to the Max wrapper class.*
- void \* **max\_jit\_obex\_new** (void \*mc, **t\_symbol** \*classname)
 

*Allocates and initializes a new Max wrapper object instance.*
- void **max\_jit\_obex\_free** (void \*x)
 

*Frees additional resources for the Max wrapper object instance.*
- **t\_jit\_err max\_jit\_obex\_attr\_set** (void \*x, **t\_symbol** \*s, long ac, **t\_atom** \*av)
 

*Sets an attribute of the Max wrapper or the wrapped Jitter object.*
- **t\_jit\_err max\_jit\_obex\_attr\_get** (void \*x, **t\_symbol** \*s, long \*ac, **t\_atom** \*\*av)
 

*Retrieves an attribute of the Max wrapper or the wrapped Jitter object.*
- void \* **max\_jit\_obex\_jitob\_get** (void \*x)
 

*Retrieves the wrapped Jitter object from a Max wrapper object.*
- void **max\_jit\_obex\_jitob\_set** (void \*x, void \*jitob)
 

*Sets the wrapped Jitter object for a Max wrapper object.*
- long **max\_jit\_obex\_inletnumber\_get** (void \*x)
 

*Retrieves the current inlet number used by inlet proxies.*
- void **max\_jit\_obex\_inletnumber\_set** (void \*x, long inletnumber)
 

*Sets the current inlet number used by inlet proxies.*
- **t\_jit\_err max\_jit\_obex\_proxy\_new** (void \*x, long c)
 

*Creates a new proxy inlet.*
- void **max\_jit\_obex\_dumpout\_set** (void \*x, void \*outlet)
 

*Sets the Max wrapper object's dump outlet's outlet pointer.*
- void \* **max\_jit\_obex\_dumpout\_get** (void \*x)

*Retrieves the Max wrapper object's dump outlet's outlet pointer.*

- void **max\_jit\_obex\_dumpout** (void \*x, **t\_symbol** \*s, short argc, **t\_atom** \*argv)

*Sends a message and arguments out the dump outlet.*

- void \* **max\_jit\_obex\_adornment\_get** (void \*x, **t\_symbol** \*classname)

*Retrieves Max wrapper object adornment specified by class name.*

- void **max\_jit\_obex\_gimmeback** (void \*x, **t\_symbol** \*s, long ac, **t\_atom** \*av)

*Calls gimmeback methods and frees any return value.*

- void **max\_jit\_obex\_gimmeback\_dumpout** (void \*x, **t\_symbol** \*s, long ac, **t\_atom** \*av)

*Calls gimmeback methods and outputs any return value through the Max wrapper class' dump outlet.*

### 38.77.1 Detailed Description

### 38.77.2 Function Documentation

#### 38.77.2.1 max\_addmethod\_defer()

```
void max_addmethod_defer (
    method m,
    char * s )
```

Adds method to Max class that calls defer rather than the method directly.

To prevent sequencing problems which arize through the use of defer, rather than defer\_low, you should instead use the max\_addmethod\_defer\_low function.

#### Parameters

<i>m</i>	method (function pointer)
<i>s</i>	method name

References A\_CANT, A\_GIMME, and addmess().

Referenced by max\_jit\_classex\_addattr().

Here is the call graph for this function:



### 38.77.2.2 max\_addmethod\_defer\_low()

```
void max_addmethod_defer_low (
    method m,
    char * s )
```

Adds method to Max class that calls defer\_low rather than the method directly.

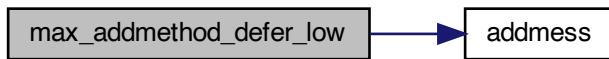
#### Parameters

<i>m</i>	method (function pointer)
<i>s</i>	method name

References A\_CANT, A\_GIMME, and addmess().

Referenced by max\_jit\_classex\_addattr(), max\_jit\_classex\_mop\_wrap(), and max\_jit\_classex\_standard\_wrap().

Here is the call graph for this function:



### 38.77.2.3 max\_addmethod\_usurp()

```
void max_addmethod_usurp (
    method m,
    char * s )
```

Adds method to Max class that uses the usurp mechanism to execute method at low priority without backlog.

Equivalent to max\_addmethod\_usurp\_low function.

#### Parameters

<i>m</i>	method (function pointer)
<i>s</i>	method name

References A\_CANT, A\_GIMME, and addmess().

Referenced by max\_jit\_classex\_addattr().

Here is the call graph for this function:



#### 38.77.2.4 max\_addmethod\_usurp\_low()

```
void max_addmethod_usurp_low (
    method m,
    char * s )
```

Adds method to Max class that uses the usurp mechanism to execute method at low priority without backlog.

##### Parameters

<i>m</i>	method (function pointer)
<i>s</i>	method name

References A\_CANT, A\_GIMME, and addmess().

Referenced by max\_jit\_classex\_addattr(), and max\_jit\_classex\_mop\_wrap().

Here is the call graph for this function:



### 38.77.2.5 max\_jit\_attr\_args()

```
void max_jit_attr_args (
    void * x,
    short ac,
    t_atom * av )
```

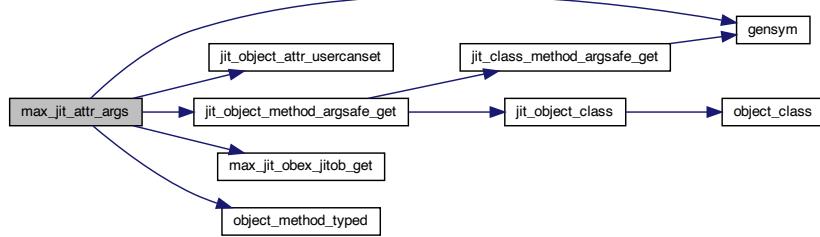
Processes attribute arguments.

#### Parameters

x	Max wrapper object pointer
ac	argument count
av	argument vector

References \_jit\_sym\_nothing, A\_SYM, gensym(), jit\_object\_attr\_usercanset(), jit\_object\_method\_argsafe\_get(), max\_jit\_obex\_jitob\_get(), object\_method\_typed(), and t\_symbol::s\_name.

Here is the call graph for this function:



### 38.77.2.6 max\_jit\_attr\_args\_offset()

```
long max_jit_attr_args_offset (
    short ac,
    t_atom * av )
```

Determines argument offset to first attribute argument.

#### Parameters

ac	argument count
av	argument vector

**Returns**

argument offset

References attr\_args\_offset().

Referenced by max\_jit\_mop\_matrix\_args().

Here is the call graph for this function:

**38.77.2.7 max\_jit\_attr\_get()**

```
t_jit_err max_jit_attr_get (
    void * x,
    t_symbol * s,
    long * ac,
    t_atom ** av )
```

Retrieves attribute value.

**Parameters**

<i>x</i>	Max wrapper object pointer
<i>s</i>	attribute name
<i>ac</i>	pointer atom count
<i>av</i>	pointer atom vector

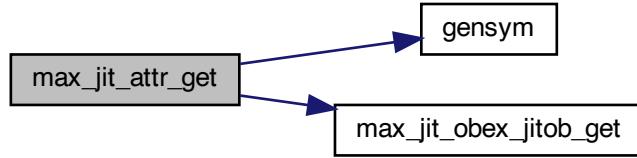
**Returns**

t\_jit\_err error code

References gensym(), max\_jit\_obex jitob\_get(), and t\_symbol::s\_name.

Referenced by max\_jit\_attr\_getdump().

Here is the call graph for this function:



### 38.77.2.8 max\_jit\_attr\_getdump()

```
void max_jit_attr_getdump (
    void * x,
    t_symbol * s,
    short argc,
    t_atom * argv )
```

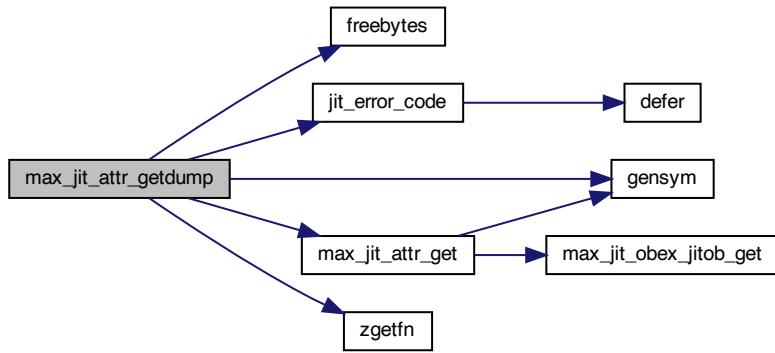
Retrieves attribute value and sends out dump outlet.

#### Parameters

<i>x</i>	Max wrapper object pointer
<i>s</i>	attribute name
<i>argc</i>	argument count (ignored)
<i>argv</i>	argument vector (ignored)

References freebytes(), gensym(), jit\_error\_code(), max\_jit\_attr\_get(), t\_symbol::s\_name, and zgetfn().

Here is the call graph for this function:



### 38.77.2.9 max\_jit\_attr\_set()

```
void max_jit_attr_set (
    void * x,
    t_symbol * s,
    short ac,
    t_atom * av )
```

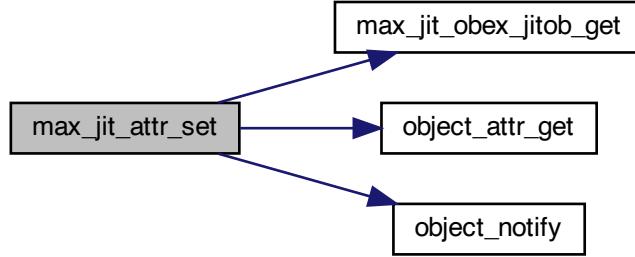
Sets attribute value.

#### Parameters

<i>x</i>	Max wrapper object pointer
<i>s</i>	attribute name
<i>ac</i>	atom count
<i>av</i>	atom vector

References max\_jit\_obex\_jitob\_get(), object\_attr\_get(), and object\_notify().

Here is the call graph for this function:



### 38.77.2.10 max\_jit\_classex\_addattr()

```
long max_jit_classex_addattr (
    void * x,
    void * attr )
```

Adds an attribute to the Max wrapper class.

#### Parameters

<i>x</i>	pointer to t_max_jit_classex data (opaque)
<i>attr</i>	attribute object pointer

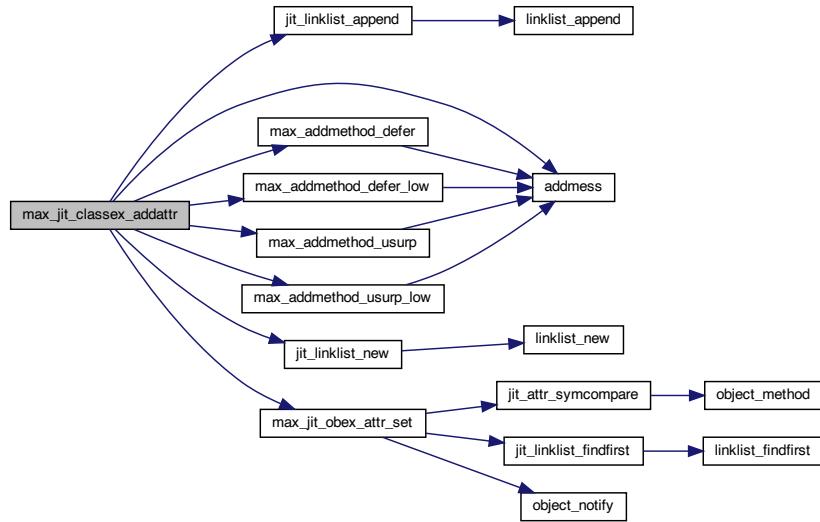
#### Returns

t\_jit\_err error code

References \_jit\_sym\_get, \_jit\_sym\_getmethod, \_jit\_sym\_getname, \_jit\_sym\_set, A\_DEFER, A\_DEFER\_LOW, A\_GIMME, A\_USURP, A\_USURP\_LOW, addmess(), jit\_linklist\_append(), jit\_linklist\_new(), max\_addmethod\_defer(), max\_addmethod\_defer\_low(), max\_addmethod\_usurp(), max\_addmethod\_usurp\_low(), max\_jit\_obex\_attr\_set(), and t\_symbol::s\_name.

Referenced by max\_jit\_classex\_mop\_wrap().

Here is the call graph for this function:



### 38.77.2.11 `max_jit_classex_setup()`

```
void * max_jit_classex_setup (
    long oboffset )
```

Allocates and initializes special t\_max\_jit\_classex data, used by the Max wrapper class.

#### Parameters

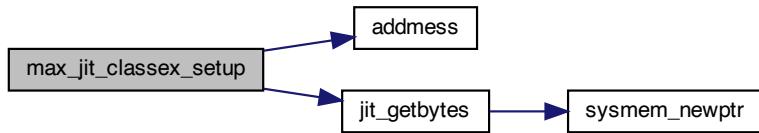
<code>oboffset</code>	object struct byte offset to obex pointer
-----------------------	---

#### Returns

pointer to t\_max\_jit\_classex data (opaque)

References A\_CANT, addmess(), and jit\_getbytes().

Here is the call graph for this function:



### 38.77.2.12 max\_jit\_classex\_standard\_wrap()

```

void max_jit_classex_standard_wrap (
    void * mclass,
    void * jclass,
    long flags )
  
```

Adds standard Jitter methods, as well as public methods and attributes of the specified Jitter class.

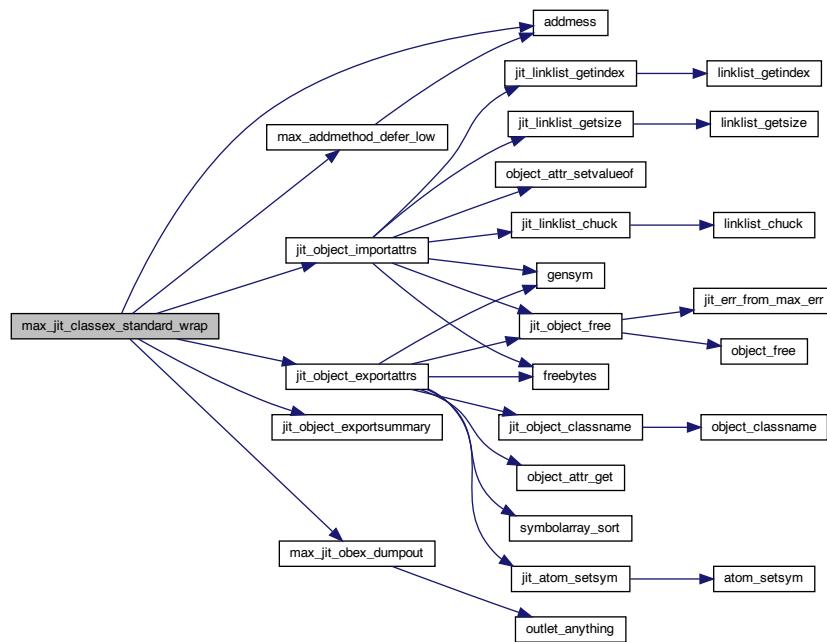
This includes the following public methods: getattributes, getstate, summary, importatrs, exportatrs; and the following private methods: dumpout, quickref, attr\_getnames, attr\_get, attr\_gettarget, and attrindex.

#### Parameters

<i>mclass</i>	Max wrapper class pointer
<i>jclass</i>	jitter class pointer
<i>flags</i>	reserved for future use (currently ignored)

References A\_CANT, addmess(), jit\_object\_exportatrs(), jit\_object\_exportsummary(), jit\_object\_importatrs(), max\_← addmethod\_defer\_low(), and max\_jit\_obex\_dumpout().

Here is the call graph for this function:



### 38.77.2.13 `max_jit_obex_adornment_get()`

```
void* max_jit_obex_adornment_get (
    void * x,
    t_symbol * classname )
```

Retrieves Max wrapper object adornment specified by class name.

Typically used for accessing the `jit_mop` adornment for MOP Max wrapper objects.

#### Parameters

<code>x</code>	Max wrapper object pointer
<code>classname</code>	adornment classname

#### Returns

adornment pointer

References `_jit_sym_findfirst`, and `jit_object_classname_compare()`.

Referenced by max\_jit\_mop\_adapt\_matrix\_all(), max\_jit\_mop\_assist(), max\_jit\_mop\_clear(), max\_jit\_mop\_free(), max\_jit\_mop\_get\_io\_by\_name(), max\_jit\_mop\_getinput(), max\_jit\_mop\_getoutput(), max\_jit\_mop\_getoutputmode(), max\_jit\_mop\_inputs(), max\_jit\_mop\_jit\_matrix(), max\_jit\_mop\_matrix\_args(), max\_jit\_mop\_matrixout\_new(), max\_jit\_mop\_notify(), max\_jit\_mop\_outputmatrix(), max\_jit\_mop\_outputs(), max\_jit\_mop\_variable\_addinputs(), and max\_jit\_mop\_variable\_addoutputs().

Here is the call graph for this function:



### 38.77.2.14 max\_jit\_obex\_attr\_get()

```
t_jit_err max_jit_obex_attr_get (
    void * x,
    t_symbol * s,
    long * ac,
    t_atom ** av )
```

Retrieves an attribute of the Max wrapper or the wrapped Jitter object.

#### Parameters

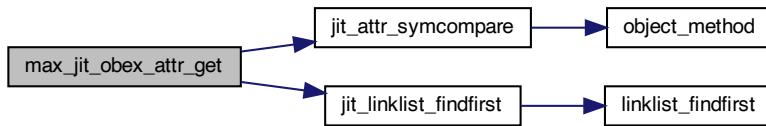
<i>x</i>	Max wrapper object pointer
<i>s</i>	attribute name
<i>ac</i>	pointer to atom count
<i>av</i>	pointer to atom vector

**Returns**

`t_jit_error` error code

References `_jit_sym_get`, `_jit_sym_getmethod`, `jit_attr_symcompare()`, and `jit_linklist_findfirst()`.

Here is the call graph for this function:

**38.77.2.15 max\_jit\_obex\_attr\_set()**

```
t_jit_err max_jit_obex_attr_set (
    void * x,
    t_symbol * s,
    long ac,
    t_atom * av )
```

Sets an attribute of the Max wrapper or the wrapped Jitter object.

**Parameters**

<code>x</code>	Max wrapper object pointer
<code>s</code>	attribute name
<code>ac</code>	atom count
<code>av</code>	atom vector

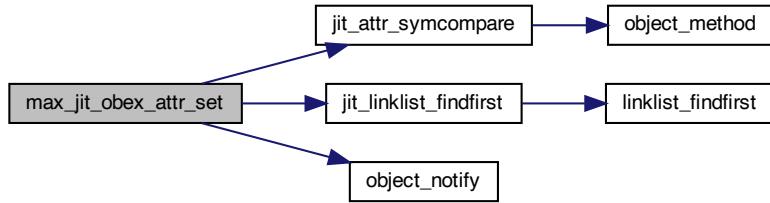
**Returns**

`t_jit_error` error code

References `_jit_sym_getmethod`, `_jit_sym_set`, `jit_attr_symcompare()`, `jit_linklist_findfirst()`, and `object_notify()`.

Referenced by `max_jit_classex_addattr()`.

Here is the call graph for this function:



### 38.77.2.16 `max_jit_obex_dumpout()`

```

void max_jit_obex_dumpout (
    void * x,
    t_symbol * s,
    short argc,
    t_atom * argv )
  
```

Sends a message and arguments out the dump outlet.

This message is equivalent to calling `outlet_anything` with the outlet returned by `max_jit_obex_dumpout_get`.

#### Parameters

<code>x</code>	Max wrapper object pointer
<code>s</code>	message symbol
<code>argc</code>	argument count
<code>argv</code>	argument vector

References `outlet_anything()`.

Referenced by `max_jit_classex_standard_wrap()`, and `max_jit_obex_gimmeback_dumpout()`.

Here is the call graph for this function:



### 38.77.2.17 max\_jit\_obex\_dumpout\_get()

```
void* max_jit_obex_dumpout_get (
    void * x )
```

Retrieves the Max wrapper object's dump outlet's outlet pointer.

#### Parameters

x	Max wrapper object pointer
---	----------------------------

#### Returns

dump outlet pointer

### 38.77.2.18 max\_jit\_obex\_dumpout\_set()

```
void max_jit_obex_dumpout_set (
    void * x,
    void * outlet )
```

Sets the Max wrapper object's dump outlet's outlet pointer.

#### Parameters

x	Max wrapper object pointer
<i>outlet</i>	dump outlet pointer

Referenced by max\_jit\_mop\_setup\_simple(), and max\_jit\_obex\_new().

### 38.77.2.19 max\_jit\_obex\_free()

```
void max_jit_obex_free (
    void * x )
```

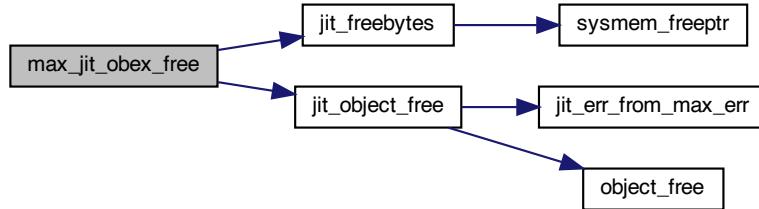
Frees additional resources for the Max wrapper object instance.

#### Parameters

x	Max wrapper object pointer
---	----------------------------

References jit\_freebytes(), and jit\_object\_free().

Here is the call graph for this function:



### 38.77.2.20 max\_jit\_obex\_gimmeback()

```
void max_jit_obex_gimmeback (
    void * x,
    t_symbol * s,
    long ac,
    t_atom * av )
```

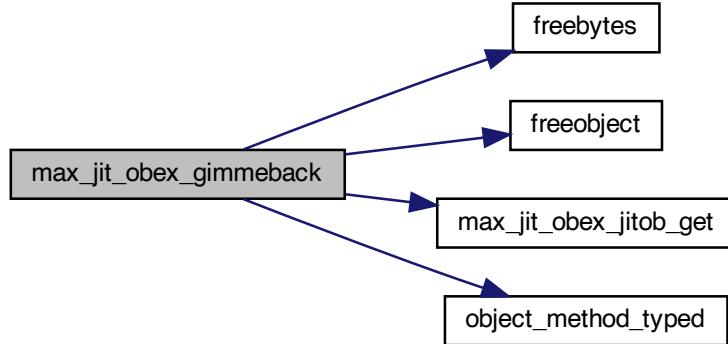
Calls gimmeback methods and frees any return value.

#### Parameters

x	Max wrapper object pointer
s	method name
ac	argument count
av	argument vector

References A NOTHING, A\_OBJ, freebytes(), freeobject(), max jit obex jitob\_get(), object\_method\_typed(), and word::w\_obj.

Here is the call graph for this function:



### 38.77.2.21 max\_jit\_obex\_gimmeback\_dumpout()

```

void max_jit_obex_gimmeback_dumpout (
    void * x,
    t_symbol * s,
    long ac,
    t_atom * av )
  
```

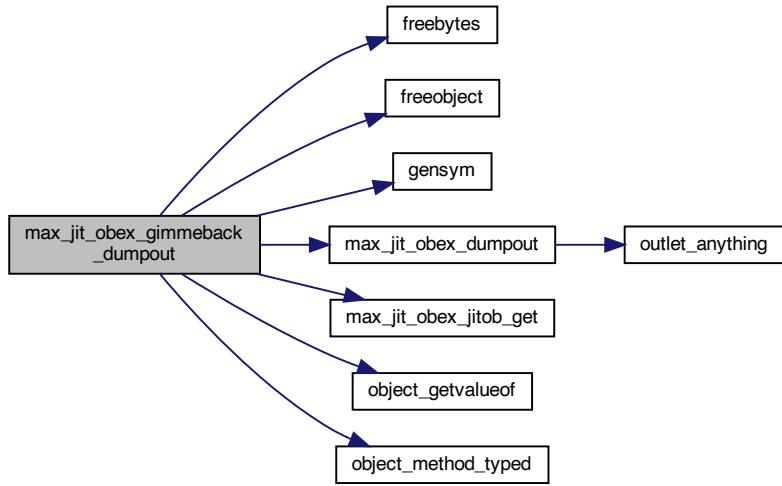
Calls gimmeback methods and outputs any return value through the Max wrapper class' dump outlet.

#### Parameters

<i>x</i>	Max wrapper object pointer
<i>s</i>	method name
<i>ac</i>	argument count
<i>av</i>	argument vector

References A NOTHING, A\_OBJ, freebytes(), freeobject(), gensym(), max jit obex dumpout(), max jit obex jitob\_get(), object\_getvalueof(), object\_method\_typed(), t\_symbol::s\_name, and word::w\_obj.

Here is the call graph for this function:



### 38.77.2.22 `max_jit_obex_inletnumber_get()`

```
long max_jit_obex_inletnumber_get (
    void * x )
```

Retrieves the current inlet number used by inlet proxies.

#### Parameters

<code>x</code>	Max wrapper object pointer
----------------	----------------------------

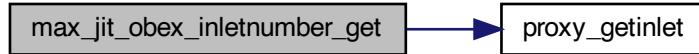
#### Returns

current inlet index

References `proxy_getinlet()`.

Referenced by `max_jit_mop_jit_matrix()`.

Here is the call graph for this function:



### 38.77.2.23 max\_jit\_obex\_inletnumber\_set()

```
void max_jit_obex_inletnumber_set (
    void * x,
    long inletnumber )
```

Sets the current inlet number used by inlet proxies.

#### Warning

Typically not used outside jitlib.

#### Parameters

x	Max wrapper object pointer
inletnumber	inlet index

Referenced by max\_jit\_obex\_new().

### 38.77.2.24 max\_jit\_obex\_jitob\_get()

```
void* max_jit_obex_jitob_get (
    void * x )
```

Retrieves the wrapped Jitter object from a Max wrapper object.

#### Parameters

x	Max wrapper object pointer
---	----------------------------

**Returns**

Jitter object pointer

Referenced by max\_jit\_attr\_args(), max\_jit\_attr\_get(), max\_jit\_attr\_set(), max\_jit\_mop\_jit\_matrix(), max\_jit\_mop\_setup(), max\_jit\_obex\_gimmeback(), and max\_jit\_obex\_gimmeback\_dumpout().

**38.77.2.25 max\_jit\_obex\_jitob\_set()**

```
void max_jit_obex_jitob_set (
    void * x,
    void * jitob )
```

Sets the wrapped Jitter object for a Max wrapper object.

**Parameters**

<i>x</i>	Max wrapper object pointer
<i>jitob</i>	Jitter object pointer

References gensym(), and OBJ\_FLAG\_REF.

Referenced by max\_jit\_mop\_setup\_simple(), and max\_jit\_obex\_new().

Here is the call graph for this function:

**38.77.2.26 max\_jit\_obex\_new()**

```
void * max_jit_obex_new (
    void * mc,
    t_symbol * classname )
```

Allocates and initializes a new Max wrapper object instance.

This is used in place of the newobject function.

**Parameters**

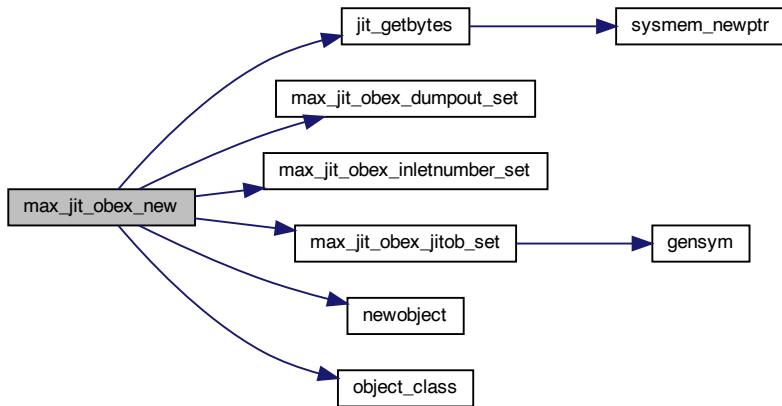
<i>mc</i>	Max class pointer
<i>classname</i>	Jitter class name to wrap

**Returns**

pointer to new Max wrapper object instance

References CLASS\_FLAG\_NOATTRIBUTES, CLASS\_FLAG\_OWNATTRIBUTES, jit\_getbytes(), max\_jit\_obex\_dumpout\_set(), max\_jit\_obex\_inletnumber\_set(), max\_jit\_obex\_jitob\_set(), newobject(), object\_class(), and t\_symbol::s\_thing.

Here is the call graph for this function:

**38.77.2.27 max\_jit\_obex\_proxy\_new()**

```
t_jit_err max_jit_obex_proxy_new (
    void * x,
    long c )
```

Creates a new proxy inlet.

**Parameters**

<i>x</i>	Max wrapper object pointer
<i>c</i>	inlet index

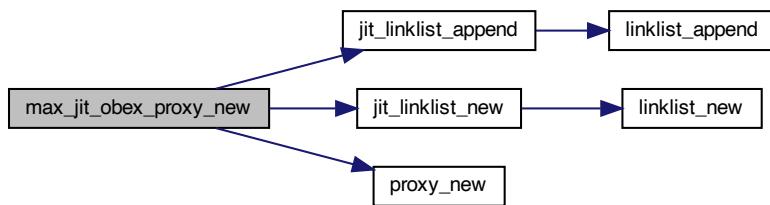
Returns

`t_jit_err` error code

References `jit_linklist_append()`, `jit_linklist_new()`, and `proxy_new()`.

Referenced by `max_jit_mop_inputs()`.

Here is the call graph for this function:



## 38.78 Memory Module

Collaboration diagram for Memory Module:



## Functions

- `void * jit_getbytes (long size)`  
*Allocates a pointer to memory.*
- `void jit_freebytes (void *ptr, long size)`  
*Frees a pointer to memory.*
- `void ** jit_handle_new (long size)`  
*Allocates a memory handle.*
- `void jit_handle_free (void **handle)`  
*Frees a memory handle.*
- `long jit_handle_size_get (void **handle)`

- Retrieves a memory handle's size in bytes.
- t\_jit\_err [jit\\_handle\\_size\\_set](#) (void \*\*handle, long size)
  - Sets a memory handle's size in bytes.
- long [jit\\_handle\\_lock](#) (void \*\*handle, long lock)
  - Sets a memory handle's lock state.
- void [jit\\_copy\\_bytes](#) (void \*dest, const void \*src, long bytes)
  - Copy bytes from source to destination pointer.
- long [jit\\_freemem](#) (void)
  - Reports free memory.
- char \* [jit\\_newptr](#) (long size)
  - Allocates a pointer to memory.
- void [jit\\_disposeptr](#) (char \*ptr)
  - Frees a pointer to memory.

### 38.78.1 Detailed Description

### 38.78.2 Function Documentation

#### 38.78.2.1 [jit\\_copy\\_bytes\(\)](#)

```
void jit_copy_bytes (
    void * dest,
    const void * src,
    long bytes )
```

Copy bytes from source to destination pointer.

#### Parameters

<i>dest</i>	destination pointer
<i>src</i>	source pointer
<i>bytes</i>	byte count to copy

References sysmem\_copyptr().

Here is the call graph for this function:



### 38.78.2.2 jit\_disposeptr()

```
void jit_disposeptr (
    char * ptr )
```

Frees a pointer to memory.

#### Warning

It is important to avoid mixing memory pools, and therefore to match calls to jit\_newptr and jit\_disposeptr.

#### Parameters

<i>ptr</i>	pointer to memory
------------	-------------------

References sysmem\_freeptr().

Here is the call graph for this function:



### 38.78.2.3 jit\_freebytes()

```
void jit_freebytes (
    void * ptr,
    long size )
```

Frees a pointer to memory.

Depending on the size of the pointer, jit\_freebytes will free from either the faster memory pool or the system memory pool.

#### Warning

It is important to avoid mixing memory pools, and therefore to match calls to jit\_getbytes and jit\_freebytes.

#### Parameters

<i>ptr</i>	pointer to memory
<i>size</i>	size in bytes allocated

References sysmem\_freeptr().

Referenced by jit\_matrix\_freedata(), and max\_jit\_obex\_free().

Here is the call graph for this function:



### 38.78.2.4 jit\_freemem()

```
long jit_freemem (
    void )
```

Reports free memory.

#### Warning

Obsolete. OS 9 only.

#### Returns

free bytes

### 38.78.2.5 jit\_getbytes()

```
void* jit_getbytes (
    long size )
```

Allocates a pointer to memory.

Depending on the size requested, jit\_getbytes will allocate from either the faster memory pool or the system memory pool.

#### Warning

It is important to avoid mixing memory pools, and therefore to match calls to jit\_getbytes and jit\_freebytes.

#### Parameters

size	size in bytes to allocate
------	---------------------------

#### Returns

pointer to memory

References sysmem\_newptr().

Referenced by jit\_matrix\_getcell(), max\_jit\_classex\_setup(), and max\_jit\_obex\_new().

Here is the call graph for this function:



### 38.78.2.6 jit\_handle\_free()

```
void jit_handle_free (
    void ** handle )
```

Frees a memory handle.

#### Warning

It is important to avoid mixing memory pools, and therefore to match calls to jit\_handle\_new and jit\_handle\_free.

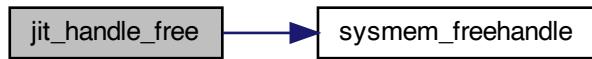
**Parameters**

<i>handle</i>	memory handle
---------------	---------------

References sysmem\_freehandle().

Referenced by jit\_matrix\_free().

Here is the call graph for this function:

**38.78.2.7 jit\_handle\_lock()**

```
long jit_handle_lock (
    void ** handle,
    long lock )
```

Sets a memory handle's lock state.

**Parameters**

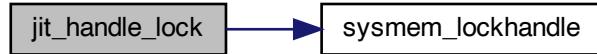
<i>handle</i>	memory handle
<i>lock</i>	state (1=locked, 0=unlocked)

**Returns**

lock state.

References sysmem\_lockhandle().

Here is the call graph for this function:



### 38.78.2.8 jit\_handle\_new()

```
void** jit_handle_new (
    long size )
```

Allocates a memory handle.

Handles are relocatable sections of memory which should be locked before dereferencing, and unlocked when not in use so that they may be relocated as necessary.

#### Warning

It is important to avoid mixing memory pools, and therefore to match calls to `jit_handle_new` and `jit_handle_free`.

#### Parameters

<code>size</code>	size in bytes to allocate
-------------------	---------------------------

#### Returns

memory handle

References `sysmem_newhandle()`.

Here is the call graph for this function:



### 38.78.2.9 jit\_handle\_size\_get()

```
long jit_handle_size_get (
    void ** handle )
```

Retrieves a memory handle's size in bytes.

#### Parameters

<i>handle</i>	memory handle
---------------	---------------

#### Returns

size in bytes

References sysmem\_handlesize().

Here is the call graph for this function:



### 38.78.2.10 jit\_handle\_size\_set()

```
t_jit_err jit_handle_size_set (
    void ** handle,
    long size )
```

Sets a memory handle's size in bytes.

#### Parameters

<i>handle</i>	memory handle
<i>size</i>	new size in bytes

**Returns**

t\_jit\_err error code.

References sysmem\_resizehandle().

Here is the call graph for this function:

**38.78.2.11 jit\_newptr()**

```
char* jit_newptr ( long size )
```

Allocates a pointer to memory.

Always allocates from the system memory pool.

**Warning**

It is important to avoid mixing memory pools, and therefore to match calls to jit\_newptr and jit\_disposeptr.

**Parameters**

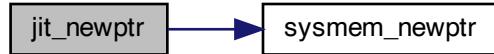
size	size in bytes to allocate
------	---------------------------

**Returns**

pointer to memory

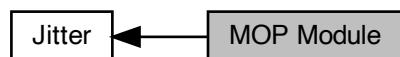
References sysmem\_newptr().

Here is the call graph for this function:



## 38.79 MOP Module

Collaboration diagram for MOP Module:



## Data Structures

- struct `t_jit_mop_io`  
`t_jit_mop_io` object struct.
- struct `t_jit_mop`  
`t_jit_mop` object struct.

## Functions

- `t_jit_object * jit_mop_io_new (void)`  
*Constructs instance of `t_jit_mop_io`.*
- `t_jit_object * jit_mop_io_newcopy (t_jit_mop_io *x)`  
*Constructs instance of `t_jit_mop_io`, copying settings of input.*
- `t_jit_err jit_mop_io_free (t_jit_mop **x)`  
*Frees instance of `t_jit_mop_io`.*
- `t_jit_err jit_mop_io_restrict_type (t_jit_mop_io *x, t_jit_matrix_info *info)`  
*Restricts the type specified in `t_jit_matrix_info` struct to those permitted by `t_jit_mop_io` instance, overwriting value in `t_jit_matrix_info` struct.*
- `t_jit_err jit_mop_io_restrict_planecount (t_jit_mop_io *x, t_jit_matrix_info *info)`

*Restricts the planecount specified in `t_jit_matrix_info` struct to those permitted by `t_jit_mop_io` instance, overwriting value in `t_jit_matrix_info` struct.*

- `t_jit_err jit_mop_io_restrict_dim (t_jit_mop_io *x, t_jit_matrix_info *info)`

*Restricts the dimension sizes specified in `t_jit_matrix_info` struct to those permitted by `t_jit_mop_io` instance, overwriting value in `t_jit_matrix_info` struct.*

- `t_jit_err jit_mop_io_matrix (t_jit_mop_io *x, void *m)`

*Sets the internal matrix reference.*

- `void * jit_mop_io_getmatrix (t_jit_mop_io *x)`

*Retrieves the internal matrix reference.*

- `t_jit_err jit_mop_io_ioproc (t_jit_mop_io *x, method ioproc)`

*Sets the I/O procedure used when handling incoming matrices.*

- `method jit_mop_io_getioproc (t_jit_mop_io *x)`

*Retrieves the I/O procedure used when handling incoming matrices.*

- `t_jit_object * jit_mop_new (long inputcount, long outputcount)`

*Constructs instance of `t_jit_mop`.*

- `t_jit_object * jit_mop_newcopy (t_jit_mop *x)`

*Constructs instance of `t_jit_mop`, copying settings of input.*

- `void * jit_mop_getinput (t_jit_mop *x, long i)`

*Retrieves input at input list index specified.*

- `void * jit_mop_getoutput (t_jit_mop *x, long i)`

*Retrieves output at output list index specified.*

- `void * jit_mop_getinputlist (t_jit_mop *x)`

*Retrieves input list.*

- `void * jit_mop_getoutputlist (t_jit_mop *x)`

*Retrieves output list.*

- `t_jit_err jit_mop_free (t_jit_mop *x)`

*Frees instance of `t_jit_mop`.*

- `t_jit_err jit_mop_single_type (void *mop, t_symbol *s)`

*Utility function to set the type attribute for all MOP inputs and outputs.*

- `t_jit_err jit_mop_single_planecount (void *mop, long c)`

*Utility function to set the planecount attribute for all MOP inputs and outputs.*

- `t_jit_err jit_mop_methodall (void *mop, t_symbol *s,...)`

*Utility function to send the same method to all MOP inputs and outputs.*

- `t_jit_err jit_mop_input_nolink (void *mop, long c)`

*Utility function to disable all linking attributes for a MOP input.*

- `t_jit_err jit_mop_output_nolink (void *mop, long c)`

*Utility function to disable all linking attributes for a MOP output.*

- `t_jit_err jit_mop_ioproc_copy_adapt (void *mop, void *mop_io, void *matrix)`

*MOP I/O procedure to copy and adapt to input.*

- `t_jit_err jit_mop_ioproc_copy_trunc (void *mop, void *mop_io, void *matrix)`

*MOP I/O procedure to copy, but truncate input.*

- `t_jit_err jit_mop_ioproc_copy_trunc_zero (void *mop, void *mop_io, void *matrix)`

*MOP I/O procedure to copy, but truncate input.*

- `t_symbol * jit_mop_ioproc_tosym (void *ioproc)`

*Utility to convert MOP I/O procedure function to a human-readable type name.*

### 38.79.1 Detailed Description

### 38.79.2 Function Documentation

#### 38.79.2.1 jit\_mop\_free()

```
t_jit_err jit_mop_free (
    t_jit_mop * x )
```

Frees instance of [t\\_jit\\_mop](#).

##### Parameters

x	<a href="#">t_jit_mop</a> object pointer
---	--

##### Returns

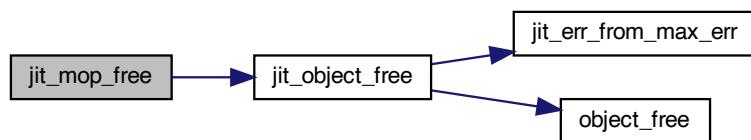
[t\\_jit\\_err](#) error code

##### Warning

Use [jit\\_object\\_free](#) instead.

References [t\\_jit\\_mop::inputlist](#), [jit\\_object\\_free\(\)](#), and [t\\_jit\\_mop::outputlist](#).

Here is the call graph for this function:



#### 38.79.2.2 jit\_mop\_getinput()

```
void * jit_mop_getinput (
    t_jit_mop * x,
    long i )
```

Retrieves input at input list index specified.

**Parameters**

<i>x</i>	<a href="#">t_jit_mop</a> object pointer
<i>i</i>	index

**Returns**[t\\_jit\\_mop\\_io](#) object pointer**Warning**

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of [t\\_jit\\_mop](#).

References [\\_jit\\_sym\\_getindex](#), and [t\\_jit\\_mop::inputlist](#).

**38.79.2.3 jit\_mop\_getinputlist()**

```
void * jit_mop_getinputlist (
    t_jit_mop * x )
```

Retrieves input list.

**Parameters**

<i>x</i>	<a href="#">t_jit_mop</a> object pointer
----------	--

**Returns**[t\\_jit\\_linklist](#) object pointer**Warning**

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of [t\\_jit\\_mop](#).

References [t\\_jit\\_mop::inputlist](#).

**38.79.2.4 jit\_mop\_getoutput()**

```
void * jit_mop_getoutput (
    t_jit_mop * x,
    long i )
```

Retrieves output at output list index specified.

**Parameters**

<i>x</i>	<a href="#">t_jit_mop</a> object pointer
<i>i</i>	index

**Returns**

[t\\_jit\\_mop\\_io](#) object pointer

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of [t\\_jit\\_mop](#).

References `_jit_sym_getindex`, and `t_jit_mop::outputlist`.

**38.79.2.5 jit\_mop\_getoutputlist()**

```
void * jit_mop_getoutputlist (
    t_jit_mop * x )
```

Retrieves output list.

**Parameters**

<i>x</i>	<a href="#">t_jit_mop</a> object pointer
----------	--

**Returns**

[t\\_jit\\_linklist](#) object pointer

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of [t\\_jit\\_mop](#).

References `t_jit_mop::outputlist`.

**38.79.2.6 jit\_mop\_input\_nolink()**

```
t_jit_err jit_mop_input_nolink (
    void * mop,
    long c )
```

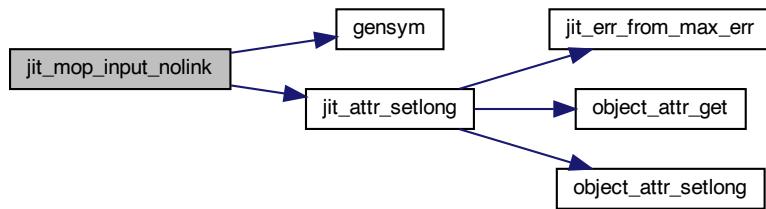
Utility function to disable all linking attributes for a MOP input.

**Parameters**

<i>mop</i>	<a href="#">t_jit_mop</a> object pointer
<i>c</i>	input index

**Returns**[t\\_jit\\_err](#) error codeReferences [\\_jit\\_sym\\_getindex](#), [gensym\(\)](#), [t\\_jit\\_mop::inputlist](#), and [jit\\_attr\\_setlong\(\)](#).

Here is the call graph for this function:

**38.79.2.7 jit\_mop\_io\_free()**

```
t_jit_err jit_mop_io_free (
    t\_jit\_mop * x )
```

Frees instance of [t\\_jit\\_mop\\_io](#).**Parameters**

<i>x</i>	<a href="#">t_jit_mop_io</a> object pointer
----------	---

**Returns**[t\\_jit\\_err](#) error code**Warning**Use [jit\\_object\\_free](#) instead.

### 38.79.2.8 jit\_mop\_io\_getioproc()

```
method jit_mop_io_getioproc (
    t_jit_mop_io * x )
```

Retrieves the I/O procedure used when handling incoming matrices.

#### Parameters

x	t_jit_mop_io object pointer
---	-----------------------------

#### Returns

I/O procedure

#### Warning

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of t\_jit\_mop\_io.

References t\_jit\_mop\_io::ioproc.

### 38.79.2.9 jit\_mop\_io\_getmatrix()

```
void * jit_mop_io_getmatrix (
    t_jit_mop_io * x )
```

Retrieves the internal matrix reference.

#### Parameters

x	t_jit_mop_io object pointer
---	-----------------------------

#### Returns

t\_jit\_matrix object pointer

#### Warning

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of t\_jit\_mop\_io.

References t\_jit\_mop\_io::matrix.

### 38.79.2.10 jit\_mop\_io\_ioproc()

```
t_jit_err jit_mop_io_ioproc (
    t_jit_mop_io * x,
    method ioproc )
```

Sets the I/O procedure used when handling incoming matrices.

#### Parameters

x	t_jit_mop_io object pointer
ioproc	I/O procedure

#### Returns

t\_jit\_err error code

#### Warning

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of [t\\_jit\\_mop\\_io](#).

References t\_jit\_mop\_io::ioproc.

### 38.79.2.11 jit\_mop\_io\_matrix()

```
t_jit_err jit_mop_io_matrix (
    t_jit_mop_io * x,
    void * m )
```

Sets the internal matrix reference.

#### Parameters

x	t_jit_mop_io object pointer
m	t_jit_matrix object pointer

#### Returns

t\_jit\_err error code

**Warning**

This function is not exported, but is provided for reference when calling via jit\_object\_method on an instance of [t\\_jit\\_mop\\_io](#).

References [t\\_jit\\_mop\\_io::matrix](#).

**38.79.2.12 jit\_mop\_io\_new()**

```
t_jit_object * jit_mop_io_new (
    void )
```

Constructs instance of [t\\_jit\\_mop\\_io](#).

**Returns**

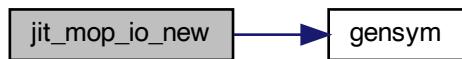
[t\\_jit\\_mop\\_io](#) object pointer

**Warning**

This function is not exported, but is provided for reference when calling via jit\_object\_new.

References [\\_jit\\_sym\\_char](#), [\\_jit\\_sym\\_float32](#), [\\_jit\\_sym\\_float64](#), [\\_jit\\_sym\\_long](#), [t\\_jit\\_mop\\_io::dimlink](#), [gensym\(\)](#), [t\\_jit\\_mop\\_io::ioname](#), [t\\_jit\\_mop\\_io::ioproc](#), [JIT\\_MATRIX\\_MAX\\_DIMCOUNT](#), [JIT\\_MATRIX\\_MAX\\_PLANECOUNT](#), [t\\_jit\\_mop\\_io::matrix](#), [t\\_jit\\_mop\\_io::matrixname](#), [t\\_jit\\_mop\\_io::maxdim](#), [t\\_jit\\_mop\\_io::maxdimcount](#), [t\\_jit\\_mop\\_io::maxplanecount](#), [t\\_jit\\_mop\\_io::mindim](#), [t\\_jit\\_mop\\_io::mindimcount](#), [t\\_jit\\_mop\\_io::minplanecount](#), [t\\_jit\\_mop\\_io::planelink](#), [t\\_jit\\_mop\\_io::special](#), [t\\_jit\\_mop\\_io::typelink](#), [t\\_jit\\_mop\\_io::types](#), and [t\\_jit\\_mop\\_io::typescount](#).

Here is the call graph for this function:

**38.79.2.13 jit\_mop\_io\_newcopy()**

```
t_jit_object * jit_mop_io_newcopy (
    t_jit_mop_io * x )
```

Constructs instance of [t\\_jit\\_mop\\_io](#), copying settings of input.

**Parameters**

<code>x</code>	<code>t_jit_mop_io</code> object pointer
----------------	--

**Returns**

`t_jit_mop_io` object pointer

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_mop_io`.

**38.79.2.14 `jit_mop_io_restrict_dim()`**

```
t_jit_err jit_mop_io_restrict_dim (
    t_jit_mop_io * x,
    t_jit_matrix_info * info )
```

Restricts the dimension sizes specified in `t_jit_matrix_info` struct to those permitted by `t_jit_mop_io` instance, overwriting value in `t_jit_matrix_info` struct.

**Parameters**

<code>x</code>	<code>t_jit_mop_io</code> object pointer
<code>info</code>	<code>t_jit_matrix_info</code> pointer

**Returns**

`t_jit_err` error code

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_mop_io`.

References `t_jit_matrix_info::dim`, `t_jit_matrix_info::dimcount`, `JIT_MATRIX_MAX_DIMCOUNT`, `MAX`, `t_jit_mop_io::maxdim`, `t_jit_mop_io::maxdimcount`, `t_jit_mop_io::mindim`, and `t_jit_mop_io::mindimcount`.

### 38.79.2.15 jit\_mop\_io\_restrict\_planeCount()

```
t_jit_err jit_mop_io_restrict_planeCount (
    t_jit_mop_io * x,
    t_jit_matrix_info * info )
```

Restricts the planeCount specified in `t_jit_matrix_info` struct to those permitted by `t_jit_mop_io` instance, overwriting value in `t_jit_matrix_info` struct.

#### Parameters

<code>x</code>	<code>t_jit_mop_io</code> object pointer
<code>info</code>	<code>t_jit_matrix_info</code> pointer

#### Returns

`t_jit_err` error code

#### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_mop_io`.

References `t_jit_mop_io::maxplaneCount`, `t_jit_mop_io::minplaneCount`, and `t_jit_matrix_info::planeCount`.

### 38.79.2.16 jit\_mop\_io\_restrict\_type()

```
t_jit_err jit_mop_io_restrict_type (
    t_jit_mop_io * x,
    t_jit_matrix_info * info )
```

Restricts the type specified in `t_jit_matrix_info` struct to those permitted by `t_jit_mop_io` instance, overwriting value in `t_jit_matrix_info` struct.

#### Parameters

<code>x</code>	<code>t_jit_mop_io</code> object pointer
<code>info</code>	<code>t_jit_matrix_info</code> pointer

#### Returns

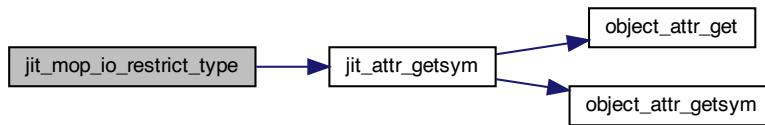
`t_jit_err` error code

### Warning

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of `t_jit_mop_io`.

References `_jit_sym_type`, `jit_attr_getsym()`, `t_jit_mop_io::matrix`, `t_jit_matrix_info::type`, `t_jit_mop_io::types`, and `t_jit_mop_io::typescount`.

Here is the call graph for this function:



### 38.79.2.17 `jit_mop_ioproc_copy_adapt()`

```
t_jit_err jit_mop_ioproc_copy_adapt (
    void * mop,
    void * mop_io,
    void * matrix )
```

MOP I/O procedure to copy and adapt to input.

#### Parameters

<code>mop</code>	<code>t_jit_mop</code> object pointer
<code>mop&lt;-_io</code>	<code>t_jit_mop_io</code> object pointer
<code>matrix</code>	<code>t_jit_matrix</code> object pointer

#### Returns

`t_jit_err` error code

```
void *m;
t_jit_matrix_info info;
if (matrix&&(m= jit_object_method(mop_io,_jit_sym_getmatrix))) {
    jit_object_method(matrix,_jit_sym_getinfo,&info);
    jit_object_method(mop_io,_jit_sym_restrict_type,&info);
    jit_object_method(mop_io,_jit_sym_restrict_dim,&info);
    jit_object_method(mop_io,_jit_sym_restrict_planeCount,&info);
    jit_object_method(m,_jit_sym_setinfo,&info);
    jit_object_method(m,_jit_sym_frommatrix,matrix,NULL);
}
```

```
return JIT_ERR_NONE;
```

References `_jit_sym_frommatrix`, `_jit_sym_getinfo`, `_jit_sym_getmatrix`, `_jit_sym_restrict_dim`, `_jit_sym_restrict_`←  
`planecount`, `_jit_sym_restrict_type`, and `_jit_sym_setinfo`.

Referenced by `jit_mop_ioproc_tosym()`.

### 38.79.2.18 `jit_mop_ioproc_copy_trunc()`

```
t_jit_err jit_mop_ioproc_copy_trunc (
    void * mop,
    void * mop_io,
    void * matrix )
```

MOP I/O procedure to copy, but truncate input.

#### Parameters

<i>mop</i>	<code>t_jit_mop</code> object pointer
<i>mop</i> ← <i>_io</i>	<code>t_jit_mop_io</code> object pointer
<i>matrix</i>	<code>t_jit_matrix</code> object pointer

#### Returns

`t_jit_err` error code

```
void *m;
t_jit_matrix_info info;
if (matrix&&(m=jit_object_method(mop_io,_jit_sym_getmatrix))) {
    jit_object_method(m,_jit_sym_frommatrix_trunc,matrix);
}
return JIT_ERR_NONE;
```

References `_jit_sym_frommatrix_trunc`, and `_jit_sym_getmatrix`.

Referenced by `jit_mop_ioproc_tosym()`.

### 38.79.2.19 `jit_mop_ioproc_copy_trunc_zero()`

```
t_jit_err jit_mop_ioproc_copy_trunc_zero (
    void * mop,
    void * mop_io,
    void * matrix )
```

MOP I/O procedure to copy, but truncate input.

Zero elsewhere.

**Parameters**

<i>mop</i>	<a href="#">t_jit_mop</a> object pointer
<i>mop</i> <i>_io</i>	<a href="#">t_jit_mop_io</a> object pointer
<i>matrix</i>	<a href="#">t_jit_matrix</a> object pointer

**Returns**[t\\_jit\\_err](#) error code

```
void *m;
t\_jit\_matrix\_info info;
if (matrix&&(m=jit_object_method(mop_io,_jit_sym_getmatrix))) {
    jit_object_method(m,_jit_sym_clear);
    jit_object_method(m,_jit_sym_frommatrix_trunc, matrix);
}
return JIT_ERR_NONE;
```

References [\\_jit\\_sym\\_clear](#), [\\_jit\\_sym\\_frommatrix\\_trunc](#), and [\\_jit\\_sym\\_getmatrix](#).

Referenced by [jit\\_mop\\_ioproc\\_tosym\(\)](#).

**38.79.2.20 jit\_mop\_ioproc\_tosym()**

```
t\_symbol* jit_mop_ioproc_tosym (
    void * ioproc )
```

Utility to convert MOP I/O procedure function to a human-readable type name.

**Parameters**

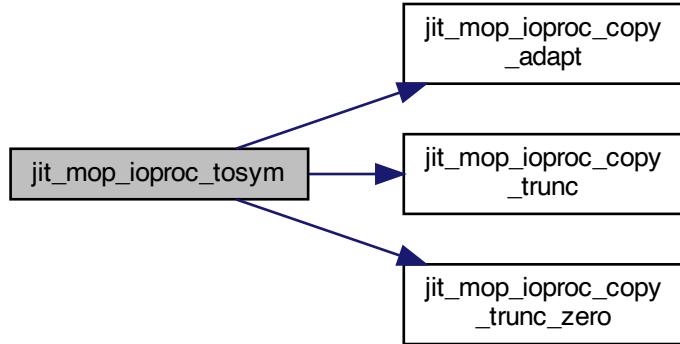
<i>ioproc</i>	<a href="#">t_jit_mop_io</a> procedure pointer
---------------	--

**Returns**[t\\_symbol](#) pointer

```
if (ioproc==NULL) {
    return ps_resamp;
} else if (ioproc==jit\_mop\_ioproc\_copy\_adapt) {
    return ps_adapt;
} else if (ioproc==jit\_mop\_ioproc\_copy\_trunc) {
    return ps_trunc;
} else if (ioproc==jit\_mop\_ioproc\_copy\_trunc\_zero) {
    return ps_trunc_zero;
} else {
    return ps_custom;
}
return ps_resamp;
```

References [jit\\_mop\\_ioproc\\_copy\\_adapt\(\)](#), [jit\\_mop\\_ioproc\\_copy\\_trunc\(\)](#), and [jit\\_mop\\_ioproc\\_copy\\_trunc\\_zero\(\)](#).

Here is the call graph for this function:



### 38.79.2.21 jit\_mop\_methodall()

```
t_jit_err jit_mop_methodall (
    void * mop,
    t_symbol * s,
    ...
)
```

Utility function to send the same method to all MOP inputs and outputs.

#### Parameters

<i>mop</i>	<a href="#">t_jit_mop</a> object pointer
<i>s</i>	method symbol
...	untyped arguments

#### Returns

*t\_jit\_err* error code

References `_jit_sym_getindex`, `t_jit_mop::inputcount`, `t_jit_mop::inputlist`, `t_jit_mop::outputcount`, and `t_jit_mop::outputlist`.

### 38.79.2.22 jit\_mop\_new()

```
t_jit_object * jit_mop_new (
    long inputcount,
    long outputcount )
```

Constructs instance of [t\\_jit\\_mop](#).

#### Returns

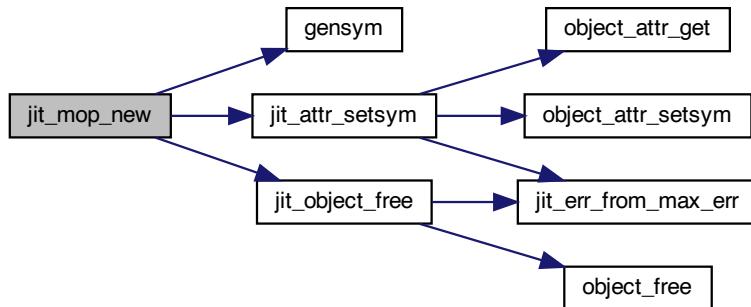
[t\\_jit\\_mop](#) object pointer

#### Warning

This function is not exported, but is provided for reference when calling via `jit_object_new`.

References `_jit_sym_append`, `t_jit_mop::adapt`, `t_jit_mop::caninplace`, `gensym()`, `t_jit_mop::inputcount`, `t_jit_mop::inputlist`, `jit_attr_setsym()`, `jit_object_free()`, `t_jit_mop::outputcount`, `t_jit_mop::outputlist`, `t_jit_mop::outputmode`, and `t_jit_mop::special`.

Here is the call graph for this function:



### 38.79.2.23 jit\_mop\_newcopy()

```
t_jit_object * jit_mop_newcopy (
    t_jit_mop * x )
```

Constructs instance of [t\\_jit\\_mop](#), copying settings of input.

**Parameters**

x	<a href="#">t_jit_mop</a> object pointer
---	--

**Returns**

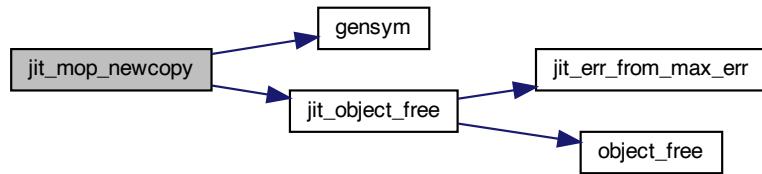
[t\\_jit\\_mop](#) object pointer

**Warning**

This function is not exported, but is provided for reference when calling via `jit_object_method` on an instance of [t\\_jit\\_mop](#).

References `_jit_sym_append`, `_jit_sym_getindex`, `_jit_sym_newcopy`, `gensym()`, `t_jit_mop::inputcount`, `t_jit_mop::inputlist`, `jit_object_free()`, `t_jit_mop::outputcount`, and `t_jit_mop::outputlist`.

Here is the call graph for this function:

**38.79.2.24 jit\_mop\_output\_nolink()**

```
t_jit_err jit_mop_output_nolink (
    void * mop,
    long c )
```

Utility function to disable all linking attributes for a MOP output.

**Parameters**

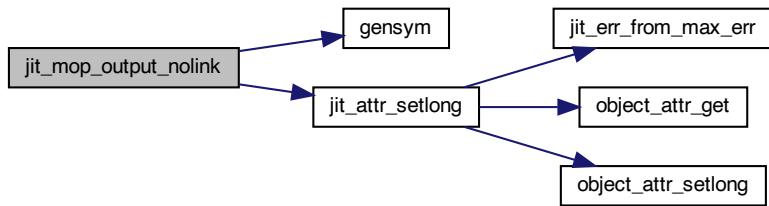
<i>mop</i>	<a href="#">t_jit_mop</a> object pointer
<i>c</i>	output index

**Returns**

`t_jit_err` error code

References `_jit_sym_getindex`, `gensym()`, `jit_attr_setlong()`, and `t_jit_mop::outputlist`.

Here is the call graph for this function:

**38.79.2.25 jit\_mop\_single\_planecount()**

```
t_jit_err jit_mop_single_planecount (
    void * mop,
    long c )
```

Utility function to set the planecount attribute for all MOP inputs and outputs.

**Parameters**

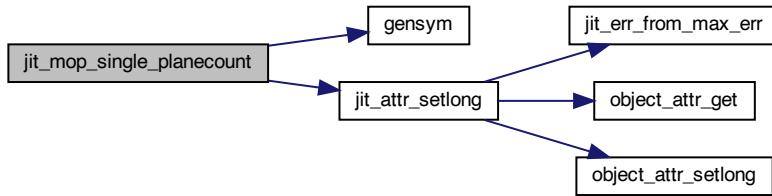
<code>mop</code>	<code>t_jit_mop</code> object pointer
<code>c</code>	planecount

**Returns**

`t_jit_err` error code

References `_jit_sym_getindex`, `gensym()`, `t_jit_mop::inputcount`, `t_jit_mop::inputlist`, `jit_attr_setlong()`, `t_jit_mop::outputcount`, and `t_jit_mop::outputlist`.

Here is the call graph for this function:



### 38.79.2.26 jit\_mop\_single\_type()

```
t_jit_err jit_mop_single_type (
    void * mop,
    t_symbol * s )
```

Utility function to set the type attribute for all MOP inputs and outputs.

#### Parameters

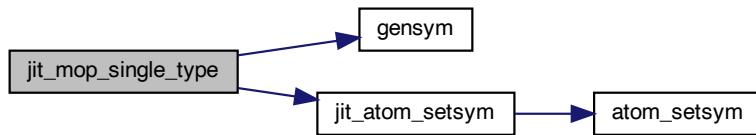
<i>mop</i>	<a href="#">t_jit_mop</a> object pointer
<i>s</i>	type symbol

#### Returns

[t\\_jit\\_err](#) error code

References [\\_jit\\_sym\\_getindex](#), [gensym\(\)](#), [t\\_jit\\_mop::inputcount](#), [t\\_jit\\_mop::inputlist](#), [jit\\_atom\\_setsym\(\)](#), [t\\_jit\\_mop::outputcount](#), and [t\\_jit\\_mop::outputlist](#).

Here is the call graph for this function:



## 38.80 Parallel Utility Module

Collaboration diagram for Parallel Utility Module:



### Functions

- void `jit_parallel_ndim_calc` (`t_jit_parallel_ndim` \*`p`)
 

*Tasks N-dimensional matrix calcuations to multiple threads if appropriate.*
- void `jit_parallel_ndim_simplecalc1` (method `fn`, `void *data`, `long dimcount`, `long *dim`, `long planeCount`, `t_jit_matrix_info *minfo1`, `char *bp1`, `long flags1`)
 

*Tasks one input/output N-dimensional matrix calcuations to multiple threads if appropriate.*
- void `jit_parallel_ndim_simplecalc2` (method `fn`, `void *data`, `long dimcount`, `long *dim`, `long planeCount`, `t_jit_matrix_info *minfo1`, `char *bp1`, `t_jit_matrix_info *minfo2`, `char *bp2`, `long flags1`, `long flags2`)
 

*Tasks two input/output N-dimensional matrix calcuations to multiple threads if appropriate.*
- void `jit_parallel_ndim_simplecalc3` (method `fn`, `void *data`, `long dimcount`, `long *dim`, `long planeCount`, `t_jit_matrix_info *minfo1`, `char *bp1`, `t_jit_matrix_info *minfo2`, `char *bp2`, `t_jit_matrix_info *minfo3`, `char *bp3`, `long flags1`, `long flags2`, `long flags3`)
 

*Tasks three input/output N-dimensional matrix calcuations to multiple threads if appropriate.*
- void `jit_parallel_ndim_simplecalc4` (method `fn`, `void *data`, `long dimcount`, `long *dim`, `long planeCount`, `t_jit_matrix_info *minfo1`, `char *bp1`, `t_jit_matrix_info *minfo2`, `char *bp2`, `t_jit_matrix_info *minfo3`, `char *bp3`, `t_jit_matrix_info *minfo4`, `char *bp4`, `long flags1`, `long flags2`, `long flags3`, `long flags4`)
 

*Tasks four input/output N-dimensional matrix calcuations to multiple threads if appropriate.*

#### 38.80.1 Detailed Description

#### 38.80.2 Function Documentation

### 38.80.2.1 jit\_parallel\_ndim\_calc()

```
void jit_parallel_ndim_calc (
    t_jit_parallel_ndim * p )
```

Tasks N-dimensional matrix calcuations to multiple threads if appropriate.

This function is ultimately what the other parallel utility functions call after having set up the `t_jit_parallel_ndim` struct. The operation is tasked to multiple threads if all of the following conditions are met:

- multiple processors or cores are present
- parallel processing is enabled
- the size of the matrix data is larger then the parallel threshold

**Parameters**

<i>p</i>	parallel ndim calc data
----------	-------------------------

Referenced by jit\_parallel\_ndim\_simplecalc1(), jit\_parallel\_ndim\_simplecalc2(), jit\_parallel\_ndim\_simplecalc3(), and jit\_parallel\_ndim\_simplecalc4().

**38.80.2.2 jit\_parallel\_ndim\_simplecalc1()**

```
void jit_parallel_ndim_simplecalc1 (
    method fn,
    void * data,
    long dimcount,
    long * dim,
    long planeCount,
    t_jit_matrix_info * minfo1,
    char * bp1,
    long flags1 )
```

Tasks one input/output N-dimensional matrix calcuations to multiple threads if appropriate.

This function fills out the t\_jit\_parallel\_ndim struct for a one input/output N-dimensional matrix calc method, and calls jit\_parallel\_ndim\_calc. This function does not distinguish between what is an input or output.

**Parameters**

<i>fn</i>	N-dimensional matrix calc method
<i>data</i>	user defined pointer (typically object)
<i>dimcount</i>	master number of dimensions to iterate
<i>dim</i>	master pointer to dimension sizes
<i>planeCount</i>	master number of planes
<i>minfo1</i>	matrix info for first input/output
<i>bp1</i>	matrix data pointer for first input/output
<i>flags1</i>	parallel flags for first input/output

References jit\_parallel\_ndim\_calc().

Here is the call graph for this function:



### 38.80.2.3 jit\_parallel\_ndim\_simplecalc2()

```

void jit_parallel_ndim_simplecalc2 (
    method fn,
    void * data,
    long dimcount,
    long * dim,
    long planeCount,
    t_jit_matrix_info * minfo1,
    char * bp1,
    t_jit_matrix_info * minfo2,
    char * bp2,
    long flags1,
    long flags2 )

```

Tasks two input/output N-dimensional matrix calcuations to multiple threads if appropriate.

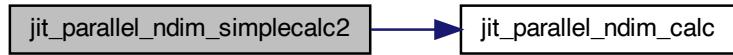
This function fills out the `t_jit_parallel_ndim` struct for a two input/output N-dimensional matrix calc method, and calls `jit_parallel_ndim_calc`. This function does not distinguish between what is an input or output.

#### Parameters

<code>fn</code>	N-dimensional matrix calc method
<code>data</code>	user defined pointer (typically object)
<code>dimcount</code>	master number of dimensions to iterate
<code>dim</code>	master pointer to dimension sizes
<code>planeCount</code>	master number of planes
<code>minfo1</code>	matrix info for first input/output
<code>bp1</code>	matrix data pointer for first input/output
<code>flags1</code>	parallel flags for first input/output
<code>minfo2</code>	matrix info for second input/output
<code>bp2</code>	matrix data pointer for second input/output
<code>flags2</code>	parallel flags for second input/output

References `t_jit_matrix_info::flags`, and `jit_parallel_ndim_calc()`.

Here is the call graph for this function:



#### 38.80.2.4 `jit_parallel_ndim_simplecalc3()`

```

void jit_parallel_ndim_simplecalc3 (
    method fn,
    void * data,
    long dimcount,
    long * dim,
    long planeCount,
    t_jit_matrix_info * minfo1,
    char * bp1,
    t_jit_matrix_info * minfo2,
    char * bp2,
    t_jit_matrix_info * minfo3,
    char * bp3,
    long flags1,
    long flags2,
    long flags3 )

```

Tasks three input/output N-dimensional matrix calcuations to multiple threads if appropriate.

This function fills out the `t_jit_parallel_ndim` struct for a three input/output N-dimensional matrix calc method, and calls `jit_parallel_ndim_calc`. This function does not distinguish between what is an input or output.

##### Parameters

<code>fn</code>	N-dimensional matrix calc method
<code>data</code>	user defined pointer (typically object)
<code>dimcount</code>	master number of dimensions to iterate
<code>dim</code>	master pointer to dimension sizes
<code>planeCount</code>	master number of planes
<code>minfo1</code>	matrix info for first input/output
<code>bp1</code>	matrix data pointer for first input/output
<code>flags1</code>	parallel flags for first input/output
<code>minfo2</code>	matrix info for second input/output

### Parameters

<i>bp2</i>	matrix data pointer for second input/output
<i>flags2</i>	parallel flags for second input/output
<i>minfo3</i>	matrix info for third input/output
<i>bp3</i>	matrix data pointer for third input/output
<i>flags3</i>	parallel flags for third input/output

References `t_jit_matrix_info::flags`, and `jit_parallel_ndim_calc()`.

Here is the call graph for this function:



### 38.80.2.5 `jit_parallel_ndim_simplecalc4()`

```

void jit_parallel_ndim_simplecalc4 (
    method fn,
    void * data,
    long dimcount,
    long * dim,
    long planeCount,
    t_jit_matrix_info * minfo1,
    char * bp1,
    t_jit_matrix_info * minfo2,
    char * bp2,
    t_jit_matrix_info * minfo3,
    char * bp3,
    t_jit_matrix_info * minfo4,
    char * bp4,
    long flags1,
    long flags2,
    long flags3,
    long flags4 )

```

Tasks four input/output N-dimensional matrix calculations to multiple threads if appropriate.

This function fills out the `t_jit_parallel_ndim` struct for a three input/output N-dimensional matrix calc method, and calls `jit_parallel_ndim_calc`. This function does not distinguish between what is an input or output.

### Parameters

<i>fn</i>	N-dimensional matrix calc method
<i>data</i>	user defined pointer (typically object)
<i>dimcount</i>	master number of dimensions to iterate
<i>dim</i>	master pointer to dimension sizes
<i>planeCount</i>	master number of planes
<i>minfo1</i>	matrix info for first input/output
<i>bp1</i>	matrix data pointer for first input/output
<i>flags1</i>	parallel flags for first input/output
<i>minfo2</i>	matrix info for second input/output
<i>bp2</i>	matrix data pointer for second input/output
<i>flags2</i>	parallel flags for second input/output
<i>minfo3</i>	matrix info for third input/output
<i>bp3</i>	matrix data pointer for third input/output
<i>flags3</i>	parallel flags for third input/output
<i>minfo4</i>	matrix info for fourth input/output
<i>bp4</i>	matrix data pointer for fourth input/output
<i>flags4</i>	parallel flags for fourth input/output

References `t_jit_matrix_info::flags`, and `jit_parallel_ndim_calc()`.

Here is the call graph for this function:



## 38.81 MOP Max Wrapper Module

Collaboration diagram for MOP Max Wrapper Module:



## Functions

- `t_jit_err max_jit_classex_mop_wrap (void *mclass, void *jclass, long flags)`  
*Adds default methods and attributes to the MOP Max wrapper class.*
- `t_jit_err max_jit_classex_mop_mproc (void *mclass, void *jclass, void *mproc)`  
*Sets a custom matrix procedure for the MOP Max wrapper class.*
- `t_jit_err max_jit_mop_setup (void *x)`  
*Sets up necessary resources for MOP Max wrapper object.*
- `t_jit_err max_jit_mop_variable_addinputs (void *x, long c)`  
*Sets the number of inputs for a variable input MOP Max wrapper object.*
- `t_jit_err max_jit_mop_variable_addoutputs (void *x, long c)`  
*Sets the number of outputs for a variable input MOP Max wrapper object.*
- `t_jit_err max_jit_mop_inputs (void *x)`  
*Creates input resources for a MOP Max wrapper object.*
- `t_jit_err max_jit_mop_outputs (void *x)`  
*Creates output resources for a MOP Max wrapper object.*
- `t_jit_err max_jit_mop_matrixout_new (void *x, long c)`  
*Creates matrix outlet for a MOP Max wrapper object.*
- `t_jit_err max_jit_mop_matrix_args (void *x, long argc, t_atom *argv)`  
*Process matrix arguments for a MOP Max wrapper object.*
- `t_jit_err max_jit_mop_jit_matrix (void *x, t_symbol *s, long argc, t_atom *argv)`  
*Default jit\_matrix method for a MOP Max wrapper object.*
- `t_jit_err max_jit_mop_assist (void *x, void *b, long m, long a, char *s)`  
*Default assist method for a MOP Max wrapper object.*
- `t_jit_err max_jit_mop_bang (void *x)`  
*Default bang method for a MOP Max wrapper object.*
- `t_jit_err max_jit_mop_outputmatrix (void *x)`  
*Default outputmatrix method for a MOP Max wrapper object.*
- `void max_jit_mop_clear (void *x)`  
*Default clear method for a MOP Max wrapper object.*
- `t_jit_err max_jit_mop_notify (void *x, t_symbol *s, t_symbol *msg)`  
*Default notify method for a MOP Max wrapper object.*
- `void max_jit_mop_free (void *x)`  
*Frees additional resources used by a MOP Max wrapper object.*
- `t_jit_err max_jit_mop_adapt_matrix_all (void *x, void *y)`  
*Adapts all input and output matrices to matrix specified.*
- `void * max_jit_mop_get_io_by_name (void *x, t_symbol *s)`  
*Retrieves `t_jit_mop_io` object pointer by name.*
- `void * max_jit_mop_getinput (void *x, long c)`  
*Retrieves input `t_jit_mop_io` object pointer index.*
- `void * max_jit_mop_getoutput (void *x, long c)`  
*Retrieves output `t_jit_mop_io` object pointer index.*
- `long max_jit_mop_getoutputmode (void *x)`  
*Retrieves current MOP Max wrapper class output mode.*
- `t_jit_err max_jit_mop_setup_simple (void *x, void *o, long argc, t_atom *argv)`  
*Initializes default state and resources for MOP Max wrapper class.*

### 38.81.1 Detailed Description

### 38.81.2 Function Documentation

#### 38.81.2.1 max\_jit\_classex\_mop\_mproc()

```
t_jit_err max_jit_classex_mop_mproc (
    void * mclass,
    void * jclass,
    void * mproc )
```

Sets a custom matrix procedure for the MOP Max wrapper class.

##### Parameters

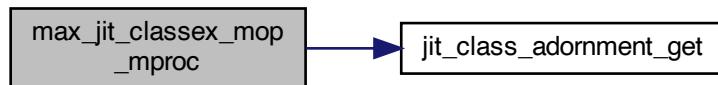
<i>mclass</i>	max jit classex pointer returned from max_jit_classex_setup
<i>jclass</i>	t_jit_class pointer, typically returned from jit_class_findbyname
<i>mproc</i>	matrix procedure

##### Returns

t\_jit\_err error code

References \_jit\_sym\_jit\_mop, \_jit\_sym\_special, and jit\_class\_adornment\_get().

Here is the call graph for this function:



#### 38.81.2.2 max\_jit\_classex\_mop\_wrap()

```
t_jit_err max_jit_classex_mop_wrap (
    void * mclass,
    void * jclass,
    long flags )
```

Adds default methods and attributes to the MOP Max wrapper class.

## Parameters

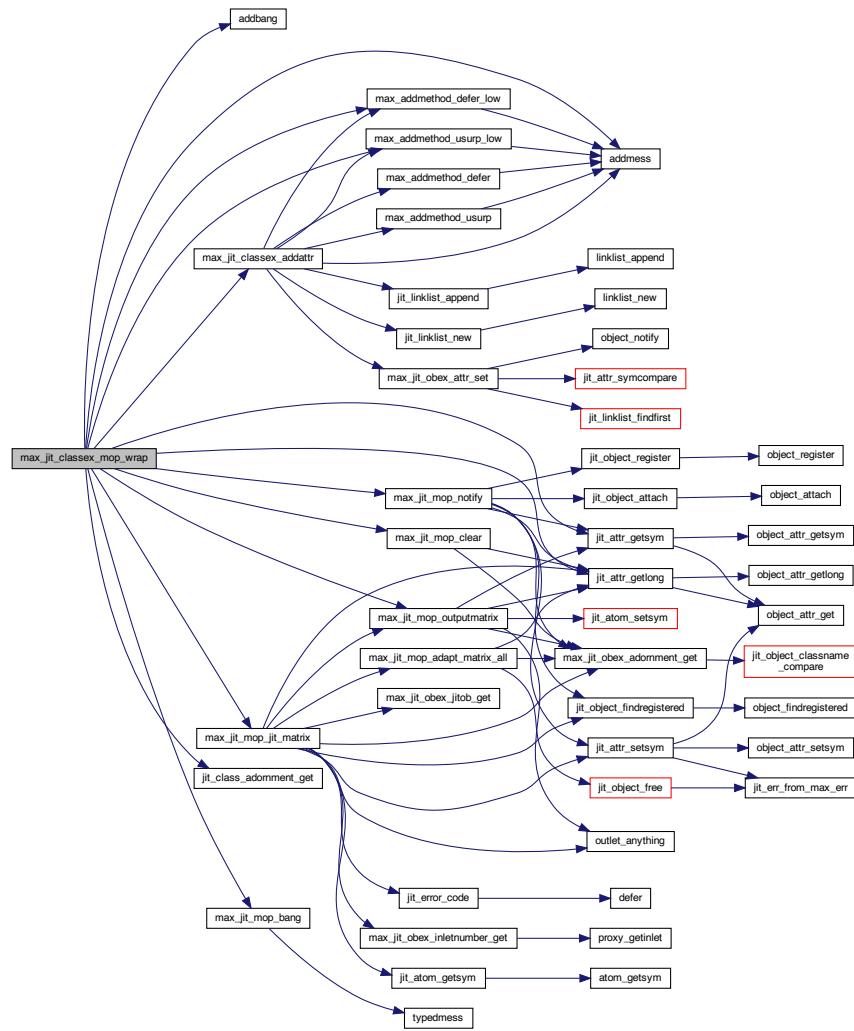
<i>mclass</i>	max jit classex pointer returned from max_jit_classex_setup
<i>jclass</i>	t_jit_class pointer, typically returned from jit_class_findbyname
<i>flags</i>	flags to override default MOP Max wrapper behavior

## Returns

`t_jit_err` error code

References `_jit_sym_atom`, `_jit_sym_char`, `_jit_sym_getinput`, `_jit_sym_getoutput`, `_jit_sym_inputcount`, `_jit_sym_ioname`, `_jit_sym_jit_attr_offset`, `_jit_sym_jit_attr_offset_array`, `_jit_sym_jit_mop`, `_jit_sym_long`, `_jit_sym_outputcount`, `_jit_sym_symbol`, `A_CANT`, `A_GIMME`, `addbang()`, `addmess()`, `JIT_ATTR_GET_DEFER_LOW`, `jit_attr_getlong()`, `jit_attr_getsym()`, `JIT_ATTR_SET_USURP_LOW`, `jit_class_adornment_get()`, `JIT_MATRIX_MAX_DIMCOUNT`, `max_addmethod_defer_low()`, `max_addmethod_usurp_low()`, `max_jit_classex_addattr()`, `max_jit_mop_bang()`, `max_jit_mop_clear()`, `MAX_JIT_MOP_FLAGS_ONLY_MATRIX_PROBE`, `MAX_JIT_MOP_FLAGS_OWN_ADAPT`, `MAX_JIT_MOP_FLAGS_OWN_BANG`, `MAX_JIT_MOP_FLAGS_OWN_CLEAR`, `MAX_JIT_MOP_FLAGS_OWN_DIM`, `MAX_JIT_MOP_FLAGS_OWN_JIT_MATRIX`, `MAX_JIT_MOP_FLAGS_OWN_NAME`, `MAX_JIT_MOP_FLAGS_OWN_NOTIFY`, `MAX_JIT_MOP_FLAGS_OWN_OUTPUTMATRIX`, `MAX_JIT_MOP_FLAGS_OWN_OUTPUTMODE`, `MAX_JIT_MOP_FLAGS_OWN_PLANECOUNT`, `MAX_JIT_MOP_FLAGS_OWN_TYPE`, `max_jit_mop_jit_matrix()`, `max_jit_mop_notify()`, `max_jit_mop_outputmatrix()`, and `t_symbol::s_name`.

Here is the call graph for this function:



### 38.81.2.3 `max_jit_mop_adapt_matrix_all()`

```
t_jit_err max_jit_mop_adapt_matrix_all (
    void * x,
    void * y )
```

Adapts all input and output matrices to matrix specified.

Typically used within the MOP Max Wrapper `jit_matrix` method for left most input.

**Parameters**

<i>x</i>	Max object pointer
<i>y</i>	matrix to adapt to

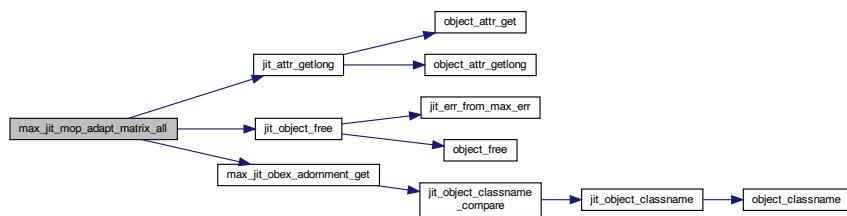
**Returns**

`t_jit_err` error code

References `_jit_sym_dimlink`, `_jit_sym_frommatrix`, `_jit_sym_getinfo`, `_jit_sym_getinput`, `_jit_sym_getioproc`, `_jit_sym_getmatrix`, `_jit_sym_getoutput`, `_jit_sym_inputcount`, `_jit_sym_jit_matrix`, `_jit_sym_jit_mop`, `_jit_sym_outputcount`, `_jit_sym_planelink`, `_jit_sym_setinfo`, `_jit_sym_typerlink`, `t_jit_matrix_info::dim`, `t_jit_matrix_info::dimcount`, `jit_attr_getlong()`, `jit_object_free()`, `max_jit_obex_adornment_get()`, `t_jit_matrix_info::planecount`, and `t_jit_matrix_info::type`.

Referenced by `max_jit_mop_jit_matrix()`.

Here is the call graph for this function:

**38.81.2.4 max\_jit\_mop\_assist()**

```
t_jit_err max_jit_mop_assist (
    void * x,
    void * b,
    long m,
    long a,
    char * s )
```

Default assist method for a MOP Max wrapper object.

**Parameters**

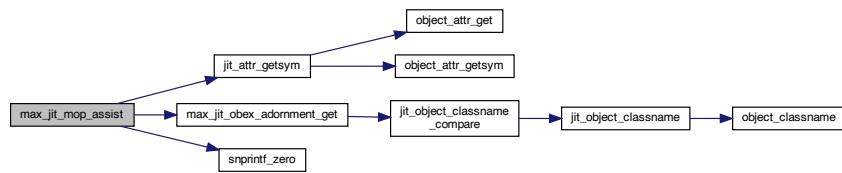
<i>x</i>	Max object pointer
<i>b</i>	ignored
<i>m</i>	inlet or outlet type
<i>a</i>	index
<i>s</i>	output string

**Returns**

`t_jit_err` error code

References `_jit_sym_getinput`, `_jit_sym_getoutput`, `_jit_sym_ioname`, `_jit_sym_jit_mop`, `jit_attr_getsym()`, `max_jit←obex_adornment_get()`, and `snprintf_zero()`.

Here is the call graph for this function:

**38.81.2.5 max\_jit\_mop\_bang()**

```
t_jit_err max_jit_mop_bang (
    void * x )
```

Default bang method for a MOP Max wrapper object.

Simply calls the default outputmatrix method.

**Parameters**

<code>x</code>	Max object pointer
----------------	--------------------

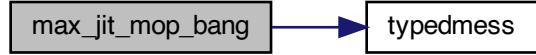
**Returns**

`t_jit_err` error code

References `_jit_sym_outputmatrix`, and `typedmess()`.

Referenced by `max_jit_classex_mop_wrap()`.

Here is the call graph for this function:



### 38.81.2.6 max\_jit\_mop\_clear()

```
void max_jit_mop_clear (
    void * x )
```

Default clear method for a MOP Max wrapper object.

Calls the clear method on all input and output matrices.

#### Parameters

x	Max object pointer
---	--------------------

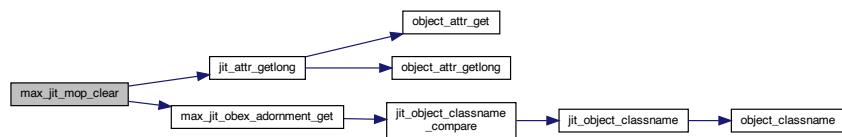
#### Returns

t\_jit\_err error code

References \_jit\_sym\_clear, \_jit\_sym\_getinput, \_jit\_sym\_getmatrix, \_jit\_sym\_getoutput, \_jit\_sym\_inputcount, \_jit\_sym←\_jit\_mop, \_jit\_sym\_outputcount, jit\_attr\_getlong(), and max\_jit\_obex\_adornment\_get().

Referenced by max\_jit\_classex\_mop\_wrap().

Here is the call graph for this function:



### 38.81.2.7 max\_jit\_mop\_free()

```
void max_jit_mop_free (
    void * x )
```

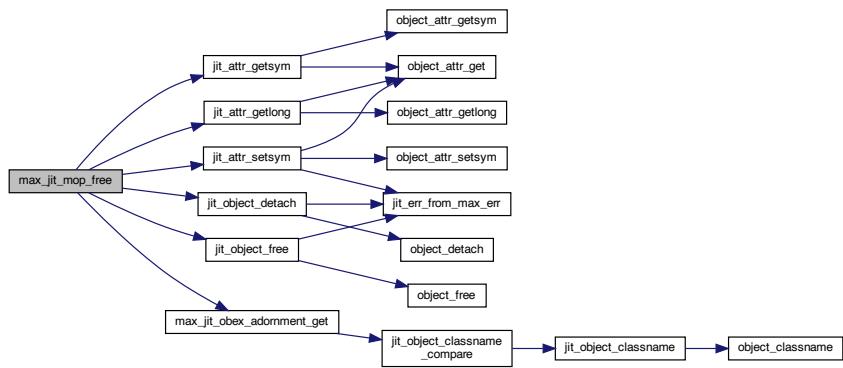
Frees additional resources used by a MOP Max wrapper object.

#### Parameters

x	Max object pointer
---	--------------------

References `_jit_sym_getinput`, `_jit_sym_getmatrix`, `_jit_sym_getoutput`, `_jit_sym_inputcount`, `_jit_sym_jit_mop`, `_jit_sym_matrix`, `_jit_sym_matrixname`, `_jit_sym_outputcount`, `jit_attr_getlong()`, `jit_attr_getsym()`, `jit_attr_setsym()`, `jit_object_detach()`, `jit_object_free()`, and `max_jit_obex_adornment_get()`.

Here is the call graph for this function:



### 38.81.2.8 max\_jit\_mop\_get\_io\_by\_name()

```
void* max_jit_mop_get_io_by_name (
    void * x,
    t_symbol * s )
```

Retrieves `t_jit_mop_io` object pointer by name.

#### Parameters

x	Max object pointer
s	input/output name (e.g. in, in2, out, out2, etc.)

**Returns**

`t_jit_err` error code

References `_jit_sym_getinput`, `_jit_sym_getoutput`, `_jit_sym_jit_mop`, `JIT_MOP_INPUT`, `JIT_MOP_OUTPUT`, and `max_jit_obex_adornment_get()`.

Here is the call graph for this function:

**38.81.2.9 `max_jit_mop_getinput()`**

```
void* max_jit_mop_getinput (
    void * x,
    long c )
```

Retrieves input `t_jit_mop_io` object pointer index.

**Parameters**

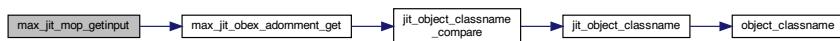
<code>x</code>	Max object pointer
<code>c</code>	input index

**Returns**

`t_jit_err` error code

References `_jit_sym_getinput`, `_jit_sym_jit_mop`, and `max_jit_obex_adornment_get()`.

Here is the call graph for this function:



### 38.81.2.10 max\_jit\_mop\_getoutput()

```
void* max_jit_mop_getoutput (
    void * x,
    long c )
```

Retrieves output [t\\_jit\\_mop\\_io](#) object pointer index.

#### Parameters

x	Max object pointer
c	output index

#### Returns

[t\\_jit\\_err](#) error code

References [\\_jit\\_sym\\_getoutput](#), [\\_jit\\_sym\\_jit\\_mop](#), and [max\\_jit\\_obex\\_adornment\\_get\(\)](#).

Here is the call graph for this function:



### 38.81.2.11 max\_jit\_mop\_getoutputmode()

```
long max_jit_mop_getoutputmode (
    void * x )
```

Retrieves current MOP Max wrapper class output mode.

#### Parameters

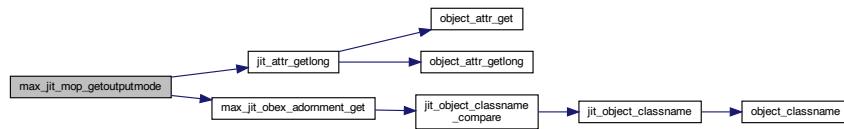
x	Max object pointer
---	--------------------

#### Returns

[t\\_jit\\_err](#) error code

References [\\_jit\\_sym\\_jit\\_mop](#), [\\_jit\\_sym\\_outputmode](#), [jit\\_attr\\_getlong\(\)](#), and [max\\_jit\\_obex\\_adornment\\_get\(\)](#).

Here is the call graph for this function:



### 38.81.2.12 `max_jit_mop_inputs()`

```
t_jit_err max_jit_mop_inputs (
    void * x )
```

Creates input resources for a MOP Max wrapper object.

#### Parameters

x	Max object pointer
---	--------------------

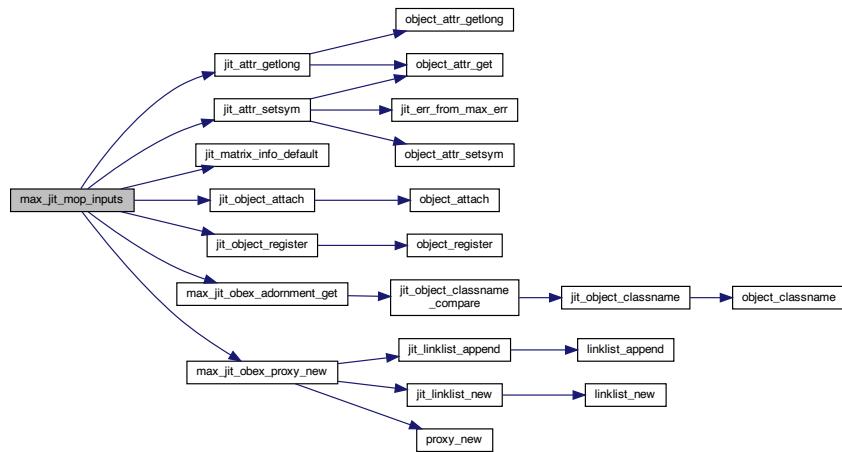
#### Returns

`t_jit_err` error code

References `_jit_sym_getinput`, `_jit_sym_inputcount`, `_jit_sym_jit_matrix`, `_jit_sym_jit_mop`, `_jit_sym_matrix`, `_jit_sym_matrixname`, `jit_attr_getlong()`, `jit_attr_setsym()`, `jit_matrix_info_default()`, `jit_object_attach()`, `jit_object_register()`, `max_jit_obex_adornment_get()`, and `max_jit_obex_proxy_new()`.

Referenced by `max_jit_mop_setup_simple()`.

Here is the call graph for this function:



### 38.81.2.13 max\_jit\_mop\_jit\_matrix()

```
t_jit_err max_jit_mop_jit_matrix (
    void * x,
    t_symbol * s,
    long argc,
    t_atom * argv )
```

Default jit\_matrix method for a MOP Max wrapper object.

#### Parameters

<i>x</i>	Max object pointer
<i>s</i>	message symbol ("jit_matrix")
<i>argc</i>	argument count
<i>argv</i>	argument vector

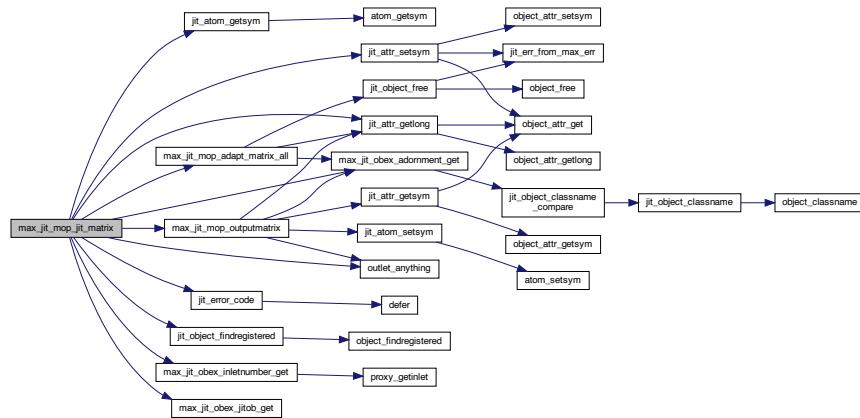
#### Returns

*t\_jit\_err* error code

References *\_jit\_sym\_adapt*, *\_jit\_sym\_class\_jit\_matrix*, *\_jit\_sym\_frommatrix*, *\_jit\_sym\_getinput*, *\_jit\_sym\_getinputlist*, *\_jit\_sym\_getioproc*, *\_jit\_sym\_getmatrix*, *\_jit\_sym\_getoutput*, *\_jit\_sym\_getoutputlist*, *\_jit\_sym\_jit\_matrix*, *\_jit\_sym\_jit\_mop*, *\_jit\_sym\_matrix*, *\_jit\_sym\_matrix\_calc*, *\_jit\_sym\_matrixname*, *\_jit\_sym\_nothing*, *\_jit\_sym\_outputmode*, *jit\_atom\_getsym()*, *jit\_attr\_getlong()*, *jit\_attr\_setsym()*, *jit\_error\_code()*, *jit\_object\_findregistered()*, *max\_jit\_mop\_adapt\_matrix\_all()*, *max\_jit\_mop\_outputmatrix()*, *max\_jit\_obex\_adornment\_get()*, *max\_jit\_obex\_inletnumber\_get()*, *max\_jit\_obex\_jitobj\_get()*, and *outlet\_anything()*.

Referenced by max\_jit\_classex\_mop\_wrap().

Here is the call graph for this function:



### **38.81.2.14 max jit mop matrix args()**

```
t_jit_err max_jit_mop_matrix_args (
    void * x,
    long argc,
    t_atom * argv )
```

Process matrix arguments for a MOP Max wrapper object.

## Parameters

<i>x</i>	Max object pointer
<i>argc</i>	argument count
<i>argv</i>	argument vector

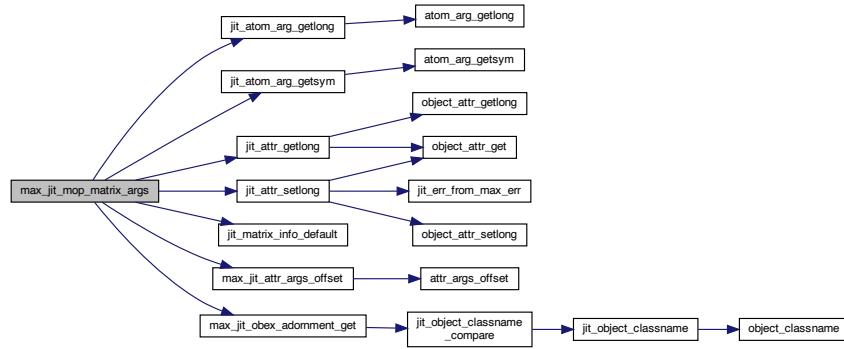
## Returns

t jit err error code

References \_jit\_sym\_adapt, \_jit\_sym\_dimlink, \_jit\_sym\_getinfo, \_jit\_sym\_getinput, \_jit\_sym\_getmatrix, \_jit\_sym\_getoutput, \_jit\_sym\_inputcount, \_jit\_sym\_jit\_mop, \_jit\_sym\_outputcount, \_jit\_sym\_outputmode, \_jit\_sym\_planelink, \_jit\_sym\_setinfo, \_jit\_sym\_typerlink, t\_jit\_matrix\_info::dim, t\_jit\_matrix\_info::dimcount, jit\_atom\_arg\_getlong(), jit\_atom\_arg\_getsym(), jit\_attr\_getlong(), jit\_attr\_setlong(), jit\_matrix\_info\_default(), max\_jit\_attr\_args\_offset(), max\_jit\_obex\_adornment\_get(), t\_jit\_matrix\_info::planecount, and t\_jit\_matrix\_info::type.

Referenced by max\_jit\_mop\_setup\_simple().

Here is the call graph for this function:



### 38.81.2.15 max\_jit\_mop\_matrixout\_new()

```
t_jit_err max_jit_mop_matrixout_new (
    void * x,
    long c )
```

Creates matrix outlet for a MOP Max wrapper object.

#### Parameters

<code>x</code>	Max object pointer
<code>c</code>	output index (zero based)

#### Returns

`t_jit_err` error code

References `_jit_sym_getoutput`, `_jit_sym_jit_mop`, `_jit_sym_special`, `max_jit_obex_adornment_get()`, and `outlet_new()`.

Referenced by `max_jit_mop_outputs()`.

Here is the call graph for this function:



### 38.81.2.16 max\_jit\_mop\_notify()

```
t_jit_err max_jit_mop_notify (
    void * x,
    t_symbol * s,
    t_symbol * msg )
```

Default notify method for a MOP Max wrapper object.

Handles any notification methods from any input and output matrix.

#### Parameters

<i>x</i>	Max object pointer
<i>s</i>	notifier name
<i>msg</i>	notification message

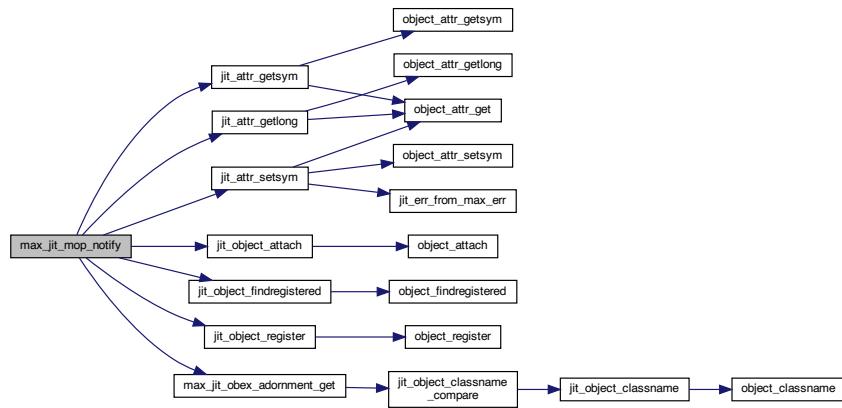
#### Returns

*t\_jit\_err* error code

References *\_jit\_sym\_free*, *\_jit\_sym\_getinfo*, *\_jit\_sym\_getinput*, *\_jit\_sym\_getoutput*, *\_jit\_sym\_inputcount*, *\_jit\_sym\_jit\_matrix*, *\_jit\_sym\_jit\_mop*, *\_jit\_sym\_matrixname*, *\_jit\_sym\_outputcount*, *jit\_attr\_getlong()*, *jit\_attr\_getsym()*, *jit\_attr\_setsym()*, *jit\_object\_attach()*, *jit\_object\_findregistered()*, *jit\_object\_register()*, and *max\_jit\_obex\_adornment\_get()*.

Referenced by *max\_jit\_classex\_mop\_wrap()*.

Here is the call graph for this function:



### 38.81.2.17 max\_jit\_mop\_outputmatrix()

```
t_jit_err max_jit_mop_outputmatrix (
    void * x )
```

Default outputmatrix method for a MOP Max wrapper object.

Calculates and outputs according to the MOP outputmode attribute.

#### Parameters

x	Max object pointer
---	--------------------

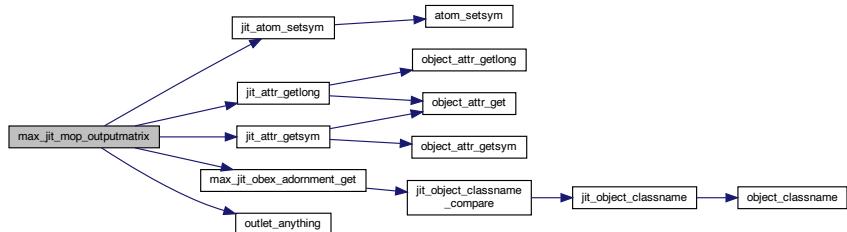
#### Returns

t\_jit\_err error code

References `_jit_sym_getoutput`, `_jit_sym_jit_matrix`, `_jit_sym_jit_mop`, `_jit_sym_matrixname`, `_jit_sym_outputcount`, `_jit_sym_outputmode`, `jit_atom_setsym()`, `jit_attr_getlong()`, `jit_attr_getsym()`, `max_jit_obex_adornment_get()`, and `outlet_anything()`.

Referenced by `max_jit_classex_mop_wrap()`, and `max_jit_mop_jit_matrix()`.

Here is the call graph for this function:



### 38.81.2.18 max\_jit\_mop\_outputs()

```
t_jit_err max_jit_mop_outputs (
    void * x )
```

Creates output resources for a MOP Max wrapper object.

**Parameters**

x	Max object pointer
---	--------------------

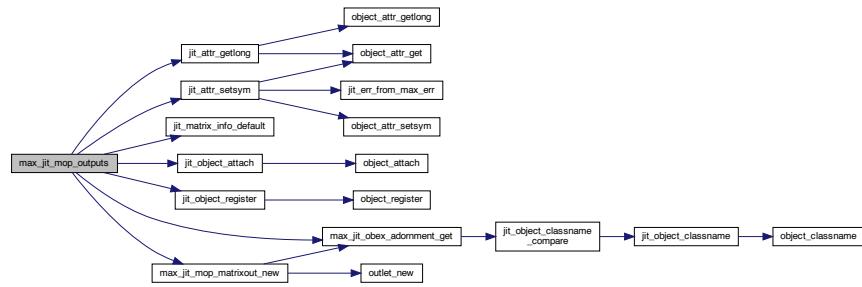
**Returns**

t\_jit\_err error code

References `_jit_sym_getoutput`, `_jit_sym_jit_matrix`, `_jit_sym_jit_mop`, `_jit_sym_matrix`, `_jit_sym_matrixname`, `_jit_sym_outputcount`, `jit_attr_getlong()`, `jit_attr_setsym()`, `jit_matrix_info_default()`, `jit_object_attach()`, `jit_object_register()`, `max_jit_mop_matrixout_new()`, and `max_jit_obex_adornment_get()`.

Referenced by `max_jit_mop_setup_simple()`.

Here is the call graph for this function:

**38.81.2.19 max\_jit\_mop\_setup()**

```
t_jit_err max_jit_mop_setup (
    void * x )
```

Sets up necessary resources for MOP Max wrapper object.

**Parameters**

x	Max object pointer
---	--------------------

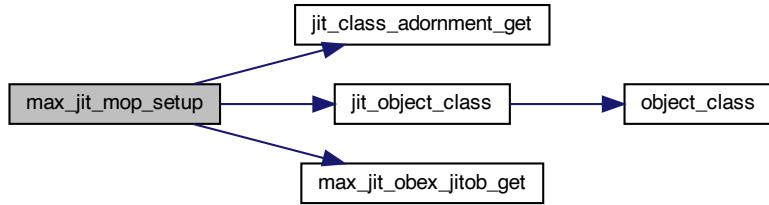
**Returns**

t\_jit\_err error code

References `_jit_sym_jit_mop`, `_jit_sym_newcopy`, `jit_class_adornment_get()`, `jit_object_class()`, and `max_jit_obex_jitob_get()`.

Referenced by max\_jit\_mop\_setup\_simple().

Here is the call graph for this function:



### 38.81.2.20 max\_jit\_mop\_setup\_simple()

```
t_jit_err max_jit_mop_setup_simple (
    void * x,
    void * o,
    long argc,
    t_atom * argv )
```

Initializes default state and resources for MOP Max wrapper class.

#### Parameters

<i>x</i>	Max object pointer
<i>o</i>	Jitter object pointer
<i>argc</i>	argument count
<i>argv</i>	argument vector

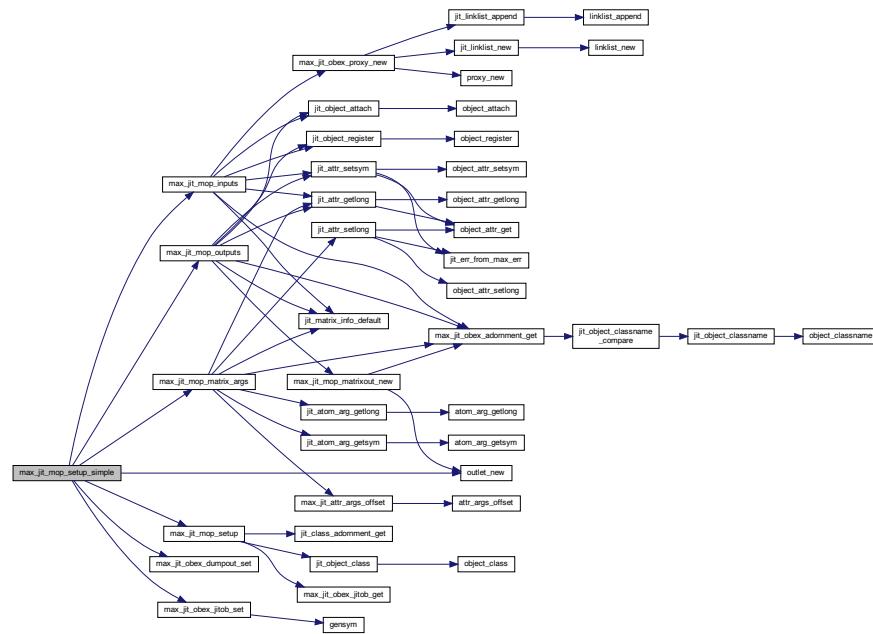
#### Returns

*t\_jit\_err* error code

```
max_jit_obex_jitob_set(x,o);
max_jit_obex_dumpout_set(x,outlet_new(x,NULL));
max_jit_mop_setup(x);
max_jit_mop_inputs(x);
max_jit_mop_outputs(x);
max_jit_mop_matrix_args(x,argc,argv);
return JIT_ERR_NONE;
```

References max\_jit\_mop\_inputs(), max\_jit\_mop\_matrix\_args(), max\_jit\_mop\_outputs(), max\_jit\_mop\_setup(), max\_jit\_obex\_dumpout\_set(), max\_jit\_obex\_jitob\_set(), and outlet\_new().

Here is the call graph for this function:



### 38.81.2.21 max\_jit\_mop\_variable\_addinputs()

```
t_jit_err max_jit_mop_variable_addinputs (
    void * x,
    long c )
```

Sets the number of inputs for a variable input MOP Max wrapper object.

#### Parameters

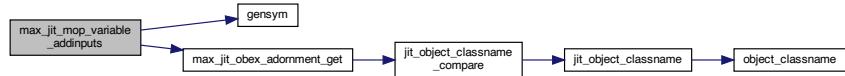
x	Max object pointer
c	inlet count

#### Returns

t\_jit\_err error code

References \_jit\_sym\_jit\_mop, gensym(), and max\_jit\_obex\_adornment\_get().

Here is the call graph for this function:



### 38.81.2.22 max\_jit\_mop\_variable\_addoutputs()

```
t_jit_err max_jit_mop_variable_addoutputs (
    void * x,
    long c )
```

Sets the number of outputs for a variable input MOP Max wrapper object.

#### Parameters

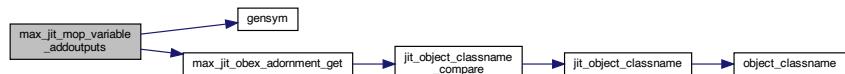
x	Max object pointer
c	inlet count

#### Returns

t\_jit\_err error code

References \_jit\_sym\_jit\_mop, gensym(), and max\_jit\_obex\_adornment\_get().

Here is the call graph for this function:



## 38.82 Operator Vector Module

Collaboration diagram for Operator Vector Module:



### Functions

- void `jit_op_vector_pass_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Unary operator: pass (char)*
- void `jit_op_vector_mult_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: multiplication (char)*
- void `jit_op_vector_div_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: division (char)*
- void `jit_op_vector_mod_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: modulo (char)*
- void `jit_op_vector_add_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: addition (char)*
- void `jit_op_vector_adds_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: saturated addition (char)*
- void `jit_op_vector_sub_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: subtraction (char)*
- void `jit_op_vector_subs_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: saturated subtraction (char)*
- void `jit_op_vector_min_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: minimum (char)*
- void `jit_op_vector_max_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: maximum (char)*
- void `jit_op_vector_avg_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: average (char)*
- void `jit_op_vector_absdiff_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: absolute difference (char)*
- void `jit_op_vector_pass_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Unary operator: pass (float32)*
- void `jit_op_vector_mult_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: multiplication (float32)*
- void `jit_op_vector_div_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: division (float32)*
- void `jit_op_vector_add_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)

- void `jit_op_vector_sub_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: addition (float32)*
  - Binary operator: subtraction (float32)*
- void `jit_op_vector_min_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: minimum (float32)*
- void `jit_op_vector_max_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: maximum (float32)*
- void `jit_op_vector_abs_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: absolute value (float32)*
- void `jit_op_vector_avg_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: average (float32)*
- void `jit_op_vector_absdiff_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: absolute difference (float32)*
- void `jit_op_vector_mod_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: modulo (float32)*
- void `jit_op_vector_fold_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: fold (float32)*
- void `jit_op_vector_wrap_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: wrap (float32)*
- void `jit_op_vector_pass_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: pass (float64)*
- void `jit_op_vector_mult_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: multiplication (float64)*
- void `jit_op_vector_div_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: division (float64)*
- void `jit_op_vector_add_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: addition (float64)*
- void `jit_op_vector_sub_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: subtraction (float64)*
- void `jit_op_vector_min_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: minimum (float64)*
- void `jit_op_vector_max_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: maximum (float64)*
- void `jit_op_vector_abs_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: absolute value (float64)*
- void `jit_op_vector_avg_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: average (float64)*
- void `jit_op_vector_absdiff_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: absolute difference (float64)*
- void `jit_op_vector_mod_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: modulo (float64)*
- void `jit_op_vector_fold_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: fold (float64)*
- void `jit_op_vector_wrap_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: wrap (float64)*
- void `jit_op_vector_pass_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: pass (long)*

- void `jit_op_vector_mult_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: multiplication (long)*
- void `jit_op_vector_div_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: division (long)*
- void `jit_op_vector_mod_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: modulo (long)*
- void `jit_op_vector_add_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: addition (long)*
- void `jit_op_vector_sub_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: subtraction (long)*
- void `jit_op_vector_min_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: minimum (long)*
- void `jit_op_vector_max_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: maximum (long)*
- void `jit_op_vector_abs_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Unary operator: absolute value (long)*
- void `jit_op_vector_avg_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: average (long)*
- void `jit_op_vector_absdiff_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: absolute difference (long)*
- void `jit_op_vector_bitand_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: bitwise and (char)*
- void `jit_op_vector_bitor_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: bitwise or (char)*
- void `jit_op_vector_bitxor_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: bitwise exclusive or (char)*
- void `jit_op_vector_bitnot_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Unary operator: bitwise not (char)*
- void `jit_op_vector_rshift_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: bitwise right shift (char)*
- void `jit_op_vector_lshift_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: bitwise left shift (char)*
- void `jit_op_vector_bitand_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: bitwise and (long)*
- void `jit_op_vector_bitor_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: bitwise or (long)*
- void `jit_op_vector_bitxor_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: bitwise exclusive or (long)*
- void `jit_op_vector_bitnot_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Unary operator: bitwise not (long)*
- void `jit_op_vector_rshift_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: bitwise right shift (long)*
- void `jit_op_vector_lshift_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Binary operator: bitwise left shift (long)*
- void `jit_op_vector_flippass_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
   
*Unary operator: flipped pass (char)*
- void `jit_op_vector_flipdiv_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)

- void `jit_op_vector_flipmod_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: flipped division (char)*
- void `jit_op_vector_flipsub_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: flipped modulo (char)*
- void `jit_op_vector_flippass_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: flipped pass (float32)*
- void `jit_op_vector_flipdiv_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: flipped division (float32)*
- void `jit_op_vector_flipmod_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: flipped modulo (float32)*
- void `jit_op_vector_flipsub_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: flipped subtraction (float32)*
- void `jit_op_vector_flippass_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: flipped pass (float64)*
- void `jit_op_vector_flipdiv_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: flipped division (float64)*
- void `jit_op_vector_flipmod_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: flipped modulo (float64)*
- void `jit_op_vector_flippass_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: flipped pass (long)*
- void `jit_op_vector_flipdiv_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: flipped division (long)*
- void `jit_op_vector_flipmod_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: flipped modulo (long)*
- void `jit_op_vector_flipsub_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: flipped subtraction (long)*
- void `jit_op_vector_and_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: logical and (char)*
- void `jit_op_vector_or_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: logical or (char)*
- void `jit_op_vector_not_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: logical not (char)*
- void `jit_op_vector_gt_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: greater than (char)*
- void `jit_op_vector_gte_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: greater than or equals (char)*
- void `jit_op_vector_lt_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: less than (char)*
- void `jit_op_vector_lte_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: less than or equals (char)*
- void `jit_op_vector_eq_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: equals (char)*
- void `jit_op_vector_neq_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: not equals (char)*
- void `jit_op_vector_and_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: logical and (float32)*

- void `jit_op_vector_or_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: logical or (float32)*
- void `jit_op_vector_not_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: logical not (float32)*
- void `jit_op_vector_gt_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: greater than (float32)*
- void `jit_op_vector_gte_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: greater than or equals (float32)*
- void `jit_op_vector_lt_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: less than (float32)*
- void `jit_op_vector_lte_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: less than or equals (float32)*
- void `jit_op_vector_eq_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: equals (float32)*
- void `jit_op_vector_neq_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: not equals (float32)*
- void `jit_op_vector_and_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: logical and (float64)*
- void `jit_op_vector_or_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: logical or (float64)*
- void `jit_op_vector_not_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: logical not (float64)*
- void `jit_op_vector_gt_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: greater than (float64)*
- void `jit_op_vector_gte_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: greater than or equals (float64)*
- void `jit_op_vector_lt_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: less than (float64)*
- void `jit_op_vector_lte_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: less than or equals (float64)*
- void `jit_op_vector_eq_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: equals (float64)*
- void `jit_op_vector_neq_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: not equals (float64)*
- void `jit_op_vector_and_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: logical and (long)*
- void `jit_op_vector_or_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: logical or (long)*
- void `jit_op_vector_not_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: logical not (long)*
- void `jit_op_vector_gt_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: greater than (long)*
- void `jit_op_vector_gte_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: greater than or equals (long)*
- void `jit_op_vector_lt_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: less than (long)*
- void `jit_op_vector_lte_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: less than or equals (long)*

- void `jit_op_vector_eq_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: less than or equals (long)*
- void `jit_op_vector_neq_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: not equals (long)*
- void `jit_op_vector_gtp_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: greater than pass (char)*
- void `jit_op_vector_gtep_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: greater than or equals pass (char)*
- void `jit_op_vector_ltp_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: less than pass (char)*
- void `jit_op_vector_ltep_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: less than or equals pass (char)*
- void `jit_op_vector_eqp_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: equals pass (char)*
- void `jit_op_vector_neqp_char` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: not equals pass (char)*
- void `jit_op_vector_gtp_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: greater than pass (float32)*
- void `jit_op_vector_gtep_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: greater than or equals pass (float32)*
- void `jit_op_vector_ltp_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: less than pass (float32)*
- void `jit_op_vector_ltep_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: less than or equals pass (float32)*
- void `jit_op_vector_eqp_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: equals pass (float32)*
- void `jit_op_vector_neqp_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: not equals pass (float32)*
- void `jit_op_vector_gtp_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: greater than pass (float64)*
- void `jit_op_vector_gtep_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: greater than or equals pass (float64)*
- void `jit_op_vector_ltp_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: less than pass (float64)*
- void `jit_op_vector_ltep_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: less than or equals pass (float64)*
- void `jit_op_vector_eqp_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: equals pass (float64)*
- void `jit_op_vector_neqp_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: not equals pass (float64)*
- void `jit_op_vector_gtp_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: greater than pass (long)*
- void `jit_op_vector_gtep_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: greater than or equals pass (long)*
- void `jit_op_vector_ltp_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
 

*Binary operator: less than pass (long)*

- void `jit_op_vector_ltep_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: less than or equals pass (long)*
- void `jit_op_vector_eqp_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: equals pass (long)*
- void `jit_op_vector_neqp_long` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: not equals pass (long)*
- void `jit_op_vector_sin_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: sine (float32)*
- void `jit_op_vector_cos_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: cosine (float32)*
- void `jit_op_vector_tan_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: tangent (float32)*
- void `jit_op_vector_asin_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: arcsine (float32)*
- void `jit_op_vector_acos_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: arccosine (float32)*
- void `jit_op_vector_atan_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: arctangent (float32)*
- void `jit_op_vector_atan2_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: arctangent (float32)*
- void `jit_op_vector_sinh_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: hyperbolic sine (float32)*
- void `jit_op_vector_cosh_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: hyperbolic cosine (float32)*
- void `jit_op_vector_tanh_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: hyperbolic tangent (float32)*
- void `jit_op_vector_asinh_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: hyperbolic arcsine (float32)*
- void `jit_op_vector_acosh_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: hyperbolic arccosine (float32)*
- void `jit_op_vector_atanh_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: hyperbolic arctangent (float32)*
- void `jit_op_vector_exp_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: exponent (float32)*
- void `jit_op_vector_exp2_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: exponent base 10 (float32)*
- void `jit_op_vector_log_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: logarithm (float32)*
- void `jit_op_vector_log2_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: logarithm base 2 (float32)*
- void `jit_op_vector_log10_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: logarithm base 10 (float32)*
- void `jit_op_vector_hypot_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: hypotenuse (float32)*
- void `jit_op_vector_pow_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: power (float32)*
- void `jit_op_vector_sqrt_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)

- void `jit_op_vector_ceil_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: ceiling (float32)*
- void `jit_op_vector_floor_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: floor (float32)*
- void `jit_op_vector_round_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: round (float32)*
- void `jit_op_vector_trunc_float32` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: truncate (float32)*
- void `jit_op_vector_sin_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: sine (float64)*
- void `jit_op_vector_cos_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: cosine (float64)*
- void `jit_op_vector_tan_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: tangent (float64)*
- void `jit_op_vector_asin_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: arcsine (float64)*
- void `jit_op_vector_acos_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: arccosine (float64)*
- void `jit_op_vector_atan_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: arctangetn (float64)*
- void `jit_op_vector_atan2_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: arctangent (float64)*
- void `jit_op_vector_sinh_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: hyperbolic sine (float64)*
- void `jit_op_vector_cosh_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: hyperbolic cosine (float64)*
- void `jit_op_vector_tanh_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: hyperbolic tangent (float64)*
- void `jit_op_vector_asinh_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: hyperbolic arcsine (float64)*
- void `jit_op_vector_acosh_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: hyperbolic arccosine (float64)*
- void `jit_op_vector_atanh_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: hyperbolic arctangent (float64)*
- void `jit_op_vector_exp_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: exponent (float64)*
- void `jit_op_vector_exp2_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: exponent base 2 (float64)*
- void `jit_op_vector_log_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: logarithm (float64)*
- void `jit_op_vector_log2_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: logarithm base 2 (float64)*
- void `jit_op_vector_log10_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Unary operator: logarithm base 10 (float64)*
- void `jit_op_vector_hypot_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)
  - Binary operator: hypotenuse (float64)*

- void `jit_op_vector_pow_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Binary operator: power (float64)*
- void `jit_op_vector_sqrt_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: square root (float64)*
- void `jit_op_vector_ceil_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: ceiling (float64)*
- void `jit_op_vector_floor_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: floor (float64)*
- void `jit_op_vector_round_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: round (float64)*
- void `jit_op_vector_trunc_float64` (long n, void \*vecdata, `t_jit_op_info` \*in0, `t_jit_op_info` \*in1, `t_jit_op_info` \*out)  
*Unary operator: truncate (float64)*

### 38.82.1 Detailed Description

### 38.82.2 Function Documentation

#### 38.82.2.1 `jit_op_vector_abs_float32()`

```
void jit_op_vector_abs_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: absolute value (float32)

##### Parameters

<code>n</code>	length of vectors
<code>vecdata</code>	ignored
<code>in0</code>	left input pointer and stride
<code>in1</code>	right input pointer and stride
<code>out</code>	output pointer and stride

#### 38.82.2.2 `jit_op_vector_abs_float64()`

```
void jit_op_vector_abs_float64 (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Unary operator: absolute value (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

#### 38.82.2.3 jit\_op\_vector\_abs\_long()

```
void jit_op_vector_abs_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: absolute value (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

#### 38.82.2.4 jit\_op\_vector\_absdiff\_char()

```
void jit_op_vector_absdiff_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: absolute difference (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.5 jit\_op\_vector\_absdiff\_float32()**

```
void jit_op_vector_absdiff_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: absolute difference (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.6 jit\_op\_vector\_absdiff\_float64()**

```
void jit_op_vector_absdiff_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: absolute difference (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.7 jit\_op\_vector\_absdiff\_long()**

```
void jit_op_vector_absdiff_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: absolute difference (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.8 jit\_op\_vector\_acos\_float32()**

```
void jit_op_vector_acos_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: arccosine (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.9 jit\_op\_vector\_acos\_float64()

```
void jit_op_vector_acos_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: arccosine (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.10 jit\_op\_vector\_acosh\_float32()

```
void jit_op_vector_acosh_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: hyperbolic arccosine (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.11 jit\_op\_vector\_acosh\_float64()

```
void jit_op_vector_acosh_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: hyperbolic arccosine (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.12 jit\_op\_vector\_add\_char()

```
void jit_op_vector_add_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: addition (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.13 jit\_op\_vector\_add\_float32()

```
void jit_op_vector_add_float32 (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: addition (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.14 jit\_op\_vector\_add\_float64()

```
void jit_op_vector_add_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: addition (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.15 jit\_op\_vector\_add\_long()

```
void jit_op_vector_add_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: addition (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.16 jit\_op\_vector\_adds\_char()**

```
void jit_op_vector_adds_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: saturated addition (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.17 jit\_op\_vector\_and\_char()**

```
void jit_op_vector_and_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: logical and (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.18 jit\_op\_vector\_and\_float32()**

```
void jit_op_vector_and_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: logical and (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.19 jit\_op\_vector\_and\_float64()**

```
void jit_op_vector_and_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: logical and (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.20 jit\_op\_vector\_and\_long()

```
void jit_op_vector_and_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: logical and (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.21 jit\_op\_vector\_asin\_float32()

```
void jit_op_vector_asin_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: arcsine (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.22 jit\_op\_vector\_asin\_float64()

```
void jit_op_vector_asin_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: arcsine (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.23 jit\_op\_vector\_asinh\_float32()

```
void jit_op_vector_asinh_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: hyperbolic arcsine (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.24 jit\_op\_vector\_asinh\_float64()

```
void jit_op_vector_asinh_float64 (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Unary operator: hyperbolic arcsine (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

#### 38.82.2.25 jit\_op\_vector\_atan2\_float32()

```
void jit_op_vector_atan2_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: arctangent (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

#### 38.82.2.26 jit\_op\_vector\_atan2\_float64()

```
void jit_op_vector_atan2_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: arctangent (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.27 jit\_op\_vector\_atan\_float32()**

```
void jit_op_vector_atan_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: arctangent (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.28 jit\_op\_vector\_atan\_float64()**

```
void jit_op_vector_atan_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: arctangetn (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.29 jit\_op\_vector\_atanh\_float32()**

```
void jit_op_vector_atanh_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: hyperbolic arctangent (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.30 jit\_op\_vector\_atanh\_float64()**

```
void jit_op_vector_atanh_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: hyperbolic arctangent (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.31 jit\_op\_vector\_avg\_char()

```
void jit_op_vector_avg_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: average (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.32 jit\_op\_vector\_avg\_float32()

```
void jit_op_vector_avg_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: average (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.33 jit\_op\_vector\_avg\_float64()**

```
void jit_op_vector_avg_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: average (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.34 jit\_op\_vector\_avg\_long()**

```
void jit_op_vector_avg_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: average (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.35 jit\_op\_vector\_bitand\_char()**

```
void jit_op_vector_bitand_char (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: bitwise and (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.36 jit\_op\_vector\_bitand\_long()

```
void jit_op_vector_bitand_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: bitwise and (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.37 jit\_op\_vector\_bitnot\_char()

```
void jit_op_vector_bitnot_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: bitwise not (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.38 jit\_op\_vector\_bitnot\_long()**

```
void jit_op_vector_bitnot_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: bitwise not (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.39 jit\_op\_vector\_bitor\_char()**

```
void jit_op_vector_bitor_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: bitwise or (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.40 jit\_op\_vector\_bitor\_long()**

```
void jit_op_vector_bitor_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: bitwise or (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.41 jit\_op\_vector\_bitxor\_char()**

```
void jit_op_vector_bitxor_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: bitwise exclusive or (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.42 jit\_op\_vector\_bitxor\_long()

```
void jit_op_vector_bitxor_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: bitwise exclusive or (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.43 jit\_op\_vector\_ceil\_float32()

```
void jit_op_vector_ceil_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: ceiling (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.44 jit\_op\_vector\_ceil\_float64()

```
void jit_op_vector_ceil_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: ceiling (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.45 jit\_op\_vector\_cos\_float32()

```
void jit_op_vector_cos_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: cosine (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.46 jit\_op\_vector\_cos\_float64()

```
void jit_op_vector_cos_float64 (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Unary operator: cosine (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

#### 38.82.2.47 jit\_op\_vector\_cosh\_float32()

```
void jit_op_vector_cosh_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: hyperbolic cosine (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

#### 38.82.2.48 jit\_op\_vector\_cosh\_float64()

```
void jit_op_vector_cosh_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: hyperbolic cosine (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.49 jit\_op\_vector\_div\_char()**

```
void jit_op_vector_div_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: division (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by jit\_op\_vector\_flipdiv\_char().

**38.82.2.50 jit\_op\_vector\_div\_float32()**

```
void jit_op_vector_div_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: division (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by `jit_op_vector_flipdiv_float32()`.

**38.82.2.51 jit\_op\_vector\_div\_float64()**

```
void jit_op_vector_div_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: division (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by `jit_op_vector_flipdiv_float64()`.

**38.82.2.52 jit\_op\_vector\_div\_long()**

```
void jit_op_vector_div_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: division (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by `jit_op_vector_flipdiv_long()`.

**38.82.2.53 jit\_op\_vector\_eq\_char()**

```
void jit_op_vector_eq_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: equals (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.54 jit\_op\_vector\_eq\_float32()**

```
void jit_op_vector_eq_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: equals (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.55 jit\_op\_vector\_eq\_float64()**

```
void jit_op_vector_eq_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: equals (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.56 jit\_op\_vector\_eq\_long()**

```
void jit_op_vector_eq_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: equals (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.57 jit\_op\_vector\_eqp\_char()**

```
void jit_op_vector_eqp_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: equals pass (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.58 jit\_op\_vector\_eqp\_float32()**

```
void jit_op_vector_eqp_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: equals pass (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.59 jit\_op\_vector\_eqp\_float64()**

```
void jit_op_vector_eqp_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: equals pass (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.60 jit\_op\_vector\_eqp\_long()**

```
void jit_op_vector_eqp_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: equals pass (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.61 jit\_op\_vector\_exp2\_float32()

```
void jit_op_vector_exp2_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: exponent base 10 (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.62 jit\_op\_vector\_exp2\_float64()

```
void jit_op_vector_exp2_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: exponent base 2(float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.63 jit\_op\_vector\_exp\_float32()

```
void jit_op_vector_exp_float32 (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Unary operator: exponent (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

#### 38.82.2.64 jit\_op\_vector\_exp\_float64()

```
void jit_op_vector_exp_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: exponent (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

#### 38.82.2.65 jit\_op\_vector\_flipdiv\_char()

```
void jit_op_vector_flipdiv_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

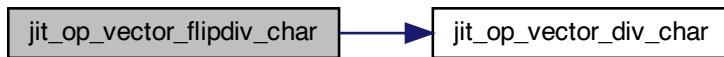
Binary operator: flipped division (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References `jit_op_vector_div_char()`.

Here is the call graph for this function:

**38.82.2.66 jit\_op\_vector\_flipdiv\_float32()**

```
void jit_op_vector_flipdiv_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

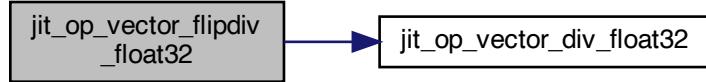
Binary operator: flipped division (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References `jit_op_vector_div_float32()`.

Here is the call graph for this function:



### 38.82.2.67 jit\_op\_vector\_flipdiv\_float64()

```
void jit_op_vector_flipdiv_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

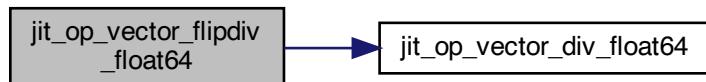
Binary operator: flipped division (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References jit\_op\_vector\_div\_float64().

Here is the call graph for this function:



### 38.82.2.68 jit\_op\_vector\_flipdiv\_long()

```
void jit_op_vector_flipdiv_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: flipped division (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References jit\_op\_vector\_div\_long().

Here is the call graph for this function:



### 38.82.2.69 jit\_op\_vector\_flipmod\_char()

```
void jit_op_vector_flipmod_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

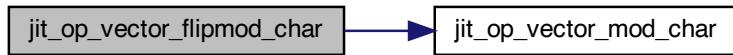
Binary operator: flipped modulo (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References `jit_op_vector_mod_char()`.

Here is the call graph for this function:

**38.82.2.70 jit\_op\_vector\_flipmod\_float32()**

```
void jit_op_vector_flipmod_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

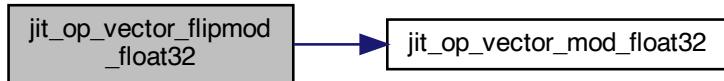
Binary operator: flipped modulo (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References `jit_op_vector_mod_float32()`.

Here is the call graph for this function:



### 38.82.2.71 jit\_op\_vector\_flipmod\_float64()

```
void jit_op_vector_flipmod_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

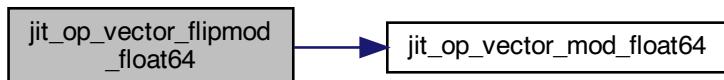
Binary operator: flipped modulo (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References jit\_op\_vector\_mod\_float64().

Here is the call graph for this function:



### 38.82.2.72 jit\_op\_vector\_flipmod\_long()

```
void jit_op_vector_flipmod_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: flipped modulo (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References jit\_op\_vector\_mod\_long().

Here is the call graph for this function:



### 38.82.2.73 jit\_op\_vector\_flippass\_char()

```
void jit_op_vector_flippass_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: flipped pass (char)

#### Parameters

<i>n</i>	length of vectors
----------	-------------------

**Parameters**

<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References `jit_op_vector_pass_char()`.

Here is the call graph for this function:

**38.82.2.74 jit\_op\_vector\_flippass\_float32()**

```
void jit_op_vector_flippass_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

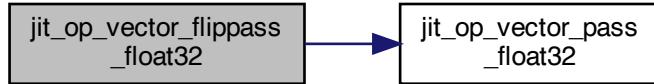
Unary operator: flipped pass (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References `jit_op_vector_pass_float32()`.

Here is the call graph for this function:



### 38.82.2.75 jit\_op\_vector\_flippass\_float64()

```
void jit_op_vector_flippass_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

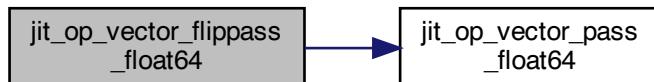
Unary operator: flipped pass (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References jit\_op\_vector\_pass\_float64().

Here is the call graph for this function:



### 38.82.2.76 jit\_op\_vector\_flippass\_long()

```
void jit_op_vector_flippass_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: flipped pass (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References jit\_op\_vector\_pass\_long().

Here is the call graph for this function:



### 38.82.2.77 jit\_op\_vector\_flipsub\_char()

```
void jit_op_vector_flipsub_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: flipped subtraction (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References `jit_op_vector_subs_char()`.

Here is the call graph for this function:



### 38.82.2.78 `jit_op_vector_flipsub_float32()`

```
void jit_op_vector_flipsub_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

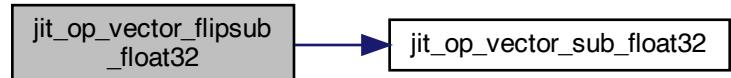
Binary operator: flipped subtraction (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References `jit_op_vector_sub_float32()`.

Here is the call graph for this function:



### 38.82.2.79 jit\_op\_vector\_flipsub\_long()

```
void jit_op_vector_flipsub_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

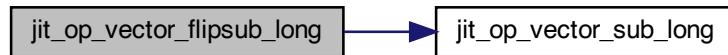
Binary operator: flipped subtraction (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References `jit_op_vector_sub_long()`.

Here is the call graph for this function:



### 38.82.2.80 jit\_op\_vector\_floor\_float32()

```
void jit_op_vector_floor_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: floor (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.81 jit\_op\_vector\_floor\_float64()**

```
void jit_op_vector_floor_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: floor (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.82 jit\_op\_vector\_fold\_float32()**

```
void jit_op_vector_fold_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: fold (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.83 jit\_op\_vector\_fold\_float64()**

```
void jit_op_vector_fold_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: fold (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.84 jit\_op\_vector\_gt\_char()**

```
void jit_op_vector_gt_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.85 jit\_op\_vector\_gt\_float32()**

```
void jit_op_vector_gt_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.86 jit\_op\_vector\_gt\_float64()**

```
void jit_op_vector_gt_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.87 jit\_op\_vector\_gt\_long()

```
void jit_op_vector_gt_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.88 jit\_op\_vector\_gte\_char()

```
void jit_op_vector_gte_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than or equals (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.89 jit\_op\_vector\_gte\_float32()

```
void jit_op_vector_gte_float32 (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: greater than or equals (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.90 jit\_op\_vector\_gte\_float64()

```
void jit_op_vector_gte_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than or equals (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.91 jit\_op\_vector\_gte\_long()

```
void jit_op_vector_gte_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than or equals (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.92 jit\_op\_vector\_gtep\_char()**

```
void jit_op_vector_gtep_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than or equals pass (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.93 jit\_op\_vector\_gtep\_float32()**

```
void jit_op_vector_gtep_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than or equals pass (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.94 jit\_op\_vector\_gtep\_float64()**

```
void jit_op_vector_gtep_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than or equals pass (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.95 jit\_op\_vector\_gtep\_long()**

```
void jit_op_vector_gtep_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than or equals pass (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.96 jit\_op\_vector\_gtp\_char()

```
void jit_op_vector_gtp_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than pass (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.97 jit\_op\_vector\_gtp\_float32()

```
void jit_op_vector_gtp_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than pass (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.98 jit\_op\_vector\_gtp\_float64()**

```
void jit_op_vector_gtp_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than pass (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.99 jit\_op\_vector\_gtp\_long()**

```
void jit_op_vector_gtp_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: greater than pass (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.100 jit\_op\_vector\_hypot\_float32()**

```
void jit_op_vector_hypot_float32 (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: hypotenuse (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.101 jit\_op\_vector\_hypot\_float64()

```
void jit_op_vector_hypot_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: hypotenuse (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.102 jit\_op\_vector\_log10\_float32()

```
void jit_op_vector_log10_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: logarithm base 10 (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.103 jit\_op\_vector\_log10\_float64()**

```
void jit_op_vector_log10_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: logarithm base 10 (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.104 jit\_op\_vector\_log2\_float32()**

```
void jit_op_vector_log2_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: logarithm base 2(float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.105 jit\_op\_vector\_log2\_float64()**

```
void jit_op_vector_log2_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: logarithm base 2 (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.106 jit\_op\_vector\_log\_float32()**

```
void jit_op_vector_log_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: logarithm (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.107 jit\_op\_vector\_log\_float64()**

```
void jit_op_vector_log_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: logarithm (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.108 jit\_op\_vector\_lshift\_char()**

```
void jit_op_vector_lshift_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: bitwise left shift (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.109 jit\_op\_vector\_lshift\_long()

```
void jit_op_vector_lshift_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: bitwise left shift (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.110 jit\_op\_vector\_lt\_char()

```
void jit_op_vector_lt_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.111 jit\_op\_vector\_lt\_float32()

```
void jit_op_vector_lt_float32 (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: less than (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.112 jit\_op\_vector\_lt\_float64()

```
void jit_op_vector_lt_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.113 jit\_op\_vector\_lt\_long()

```
void jit_op_vector_lt_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.114 jit\_op\_vector\_lte\_char()**

```
void jit_op_vector_lte_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than or equals (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.115 jit\_op\_vector\_lte\_float32()**

```
void jit_op_vector_lte_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than or equals (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.116 jit\_op\_vector\_lte\_float64()**

```
void jit_op_vector_lte_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than or equals (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.117 jit\_op\_vector\_lte\_long()**

```
void jit_op_vector_lte_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than or equals (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.118 jit\_op\_vector\_ltep\_char()

```
void jit_op_vector_ltep_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than or equals pass (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.119 jit\_op\_vector\_ltep\_float32()

```
void jit_op_vector_ltep_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than or equals pass (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.120 jit\_op\_vector\_ltep\_float64()**

```
void jit_op_vector_ltep_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than or equals pass (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.121 jit\_op\_vector\_ltep\_long()**

```
void jit_op_vector_ltep_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than or equals pass (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.122 jit\_op\_vector\_ltp\_char()**

```
void jit_op_vector_ltp_char (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: less than pass (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.123 jit\_op\_vector\_ltp\_float32()

```
void jit_op_vector_ltp_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than pass (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.124 jit\_op\_vector\_ltp\_float64()

```
void jit_op_vector_ltp_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than pass (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.125 jit\_op\_vector\_ltp\_long()**

```
void jit_op_vector_ltp_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: less than pass (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.126 jit\_op\_vector\_max\_char()**

```
void jit_op_vector_max_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: maximum (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.127 jit\_op\_vector\_max\_float32()**

```
void jit_op_vector_max_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: maximum (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.128 jit\_op\_vector\_max\_float64()**

```
void jit_op_vector_max_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: maximum (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.129 jit\_op\_vector\_max\_long()**

```
void jit_op_vector_max_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: maximum (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.130 jit\_op\_vector\_min\_char()**

```
void jit_op_vector_min_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: minimum (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.131 jit\_op\_vector\_min\_float32()

```
void jit_op_vector_min_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: minimum (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.132 jit\_op\_vector\_min\_float64()

```
void jit_op_vector_min_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: minimum (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.133 jit\_op\_vector\_min\_long()

```
void jit_op_vector_min_long (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: minimum (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

#### 38.82.2.134 jit\_op\_vector\_mod\_char()

```
void jit_op_vector_mod_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: modulo (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by jit\_op\_vector\_flipmod\_char().

#### 38.82.2.135 jit\_op\_vector\_mod\_float32()

```
void jit_op_vector_mod_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
```

```
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: modulo (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by jit\_op\_vector\_flipmod\_float32().

### 38.82.2.136 jit\_op\_vector\_mod\_float64()

```
void jit_op_vector_mod_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: modulo (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by jit\_op\_vector\_flipmod\_float64().

### 38.82.2.137 jit\_op\_vector\_mod\_long()

```
void jit_op_vector_mod_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
```

```
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: modulo (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by jit\_op\_vector\_flipmod\_long().

### 38.82.2.138 jit\_op\_vector\_mult\_char()

```
void jit_op_vector_mult_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: multiplication (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.139 jit\_op\_vector\_mult\_float32()

```
void jit_op_vector_mult_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: multiplication (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.140 jit\_op\_vector\_mult\_float64()**

```
void jit_op_vector_mult_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: multiplication (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.141 jit\_op\_vector\_mult\_long()**

```
void jit_op_vector_mult_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: multiplication (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.142 jit\_op\_vector\_neq\_char()**

```
void jit_op_vector_neq_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: not equals (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.143 jit\_op\_vector\_neq\_float32()**

```
void jit_op_vector_neq_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: not equals (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.144 jit\_op\_vector\_neq\_float64()

```
void jit_op_vector_neq_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: not equals (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.145 jit\_op\_vector\_neq\_long()

```
void jit_op_vector_neq_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: not equals (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.146 jit\_op\_vector\_neqp\_char()**

```
void jit_op_vector_neqp_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: not equals pass (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.147 jit\_op\_vector\_neqp\_float32()**

```
void jit_op_vector_neqp_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: not equals pass (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.148 jit\_op\_vector\_neqp\_float64()**

```
void jit_op_vector_neqp_float64 (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: not equals pass (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.149 jit\_op\_vector\_neqp\_long()

```
void jit_op_vector_neqp_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: not equals pass (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.150 jit\_op\_vector\_not\_char()

```
void jit_op_vector_not_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: logical not (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.151 jit\_op\_vector\_not\_float32()**

```
void jit_op_vector_not_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: logical not (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.152 jit\_op\_vector\_not\_float64()**

```
void jit_op_vector_not_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: logical not (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.153 jit\_op\_vector\_not\_long()**

```
void jit_op_vector_not_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: logical not (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.154 jit\_op\_vector\_or\_char()**

```
void jit_op_vector_or_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: logical or (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.155 jit\_op\_vector\_or\_float32()**

```
void jit_op_vector_or_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: logical or (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.156 jit\_op\_vector\_or\_float64()**

```
void jit_op_vector_or_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: logical or (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.157 jit\_op\_vector\_or\_long()

```
void jit_op_vector_or_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: logical or (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.158 jit\_op\_vector\_pass\_char()

```
void jit_op_vector_pass_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: pass (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by jit\_op\_vector\_flippass\_char().

### 38.82.2.159 jit\_op\_vector\_pass\_float32()

```
void jit_op_vector_pass_float32 (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Unary operator: pass (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by jit\_op\_vector\_flippass\_float32().

#### 38.82.2.160 jit\_op\_vector\_pass\_float64()

```
void jit_op_vector_pass_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: pass (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

References t\_jit\_op\_info::p, and t\_jit\_op\_info::stride.

Referenced by jit\_op\_vector\_flippass\_float64().

#### 38.82.2.161 jit\_op\_vector\_pass\_long()

```
void jit_op_vector_pass_long (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Unary operator: pass (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by jit\_op\_vector\_flippass\_long().

### 38.82.2.162 jit\_op\_vector\_pow\_float32()

```
void jit_op_vector_pow_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: power (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.163 jit\_op\_vector\_pow\_float64()

```
void jit_op_vector_pow_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
```

```
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: power (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.164 jit\_op\_vector\_round\_float32()

```
void jit_op_vector_round_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: round (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.165 jit\_op\_vector\_round\_float64()

```
void jit_op_vector_round_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: round (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.166 jit\_op\_vector\_rshift\_char()**

```
void jit_op_vector_rshift_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: bitwise right shift (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.167 jit\_op\_vector\_rshift\_long()**

```
void jit_op_vector_rshift_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: bitwise right shift (long)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.168 jit\_op\_vector\_sin\_float32()**

```
void jit_op_vector_sin_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: sine (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.169 jit\_op\_vector\_sin\_float64()**

```
void jit_op_vector_sin_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: sine (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.170 jit\_op\_vector\_sinh\_float32()

```
void jit_op_vector_sinh_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: hyperbolic sine (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.171 jit\_op\_vector\_sinh\_float64()

```
void jit_op_vector_sinh_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: hyperbolic sine (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.172 jit\_op\_vector\_sqrt\_float32()**

```
void jit_op_vector_sqrt_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: square root (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.173 jit\_op\_vector\_sqrt\_float64()**

```
void jit_op_vector_sqrt_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: square root (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.174 jit\_op\_vector\_sub\_char()**

```
void jit_op_vector_sub_char (
    long n,
```

```
void * vecdata,
t_jit_op_info * in0,
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: subtraction (char)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.175 jit\_op\_vector\_sub\_float32()

```
void jit_op_vector_sub_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: subtraction (float32)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by jit\_op\_vector\_flipsub\_float32().

### 38.82.2.176 jit\_op\_vector\_sub\_float64()

```
void jit_op_vector_sub_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
```

```
t_jit_op_info * in1,
t_jit_op_info * out )
```

Binary operator: subtraction (float64)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

### 38.82.2.177 jit\_op\_vector\_sub\_long()

```
void jit_op_vector_sub_long (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: subtraction (long)

#### Parameters

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by jit\_op\_vector\_flipsub\_long().

### 38.82.2.178 jit\_op\_vector\_subs\_char()

```
void jit_op_vector_subs_char (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: saturated subtraction (char)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

Referenced by jit\_op\_vector\_flipsub\_char().

**38.82.2.179 jit\_op\_vector\_tan\_float32()**

```
void jit_op_vector_tan_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: tangent (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.180 jit\_op\_vector\_tan\_float64()**

```
void jit_op_vector_tan_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: tangent (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.181 jit\_op\_vector\_tanh\_float32()**

```
void jit_op_vector_tanh_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: hyperbolic tangent (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.182 jit\_op\_vector\_tanh\_float64()**

```
void jit_op_vector_tanh_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: hyperbolic tangent (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored

**Parameters**

<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.183 jit\_op\_vector\_trunc\_float32()**

```
void jit_op_vector_trunc_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: truncate (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.184 jit\_op\_vector\_trunc\_float64()**

```
void jit_op_vector_trunc_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Unary operator: truncate (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.185 jit\_op\_vector\_wrap\_float32()**

```
void jit_op_vector_wrap_float32 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: wrap (float32)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride

**38.82.2.186 jit\_op\_vector\_wrap\_float64()**

```
void jit_op_vector_wrap_float64 (
    long n,
    void * vecdata,
    t_jit_op_info * in0,
    t_jit_op_info * in1,
    t_jit_op_info * out )
```

Binary operator: wrap (float64)

**Parameters**

<i>n</i>	length of vectors
<i>vecdata</i>	ignored
<i>in0</i>	left input pointer and stride
<i>in1</i>	right input pointer and stride
<i>out</i>	output pointer and stride



## Chapter 39

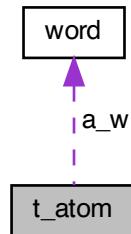
# Data Structure Documentation

### 39.1 t\_atom Struct Reference

An atom is a typed datum.

```
#include <ext_mess.h>
```

Collaboration diagram for t\_atom:



#### 39.1.1 Detailed Description

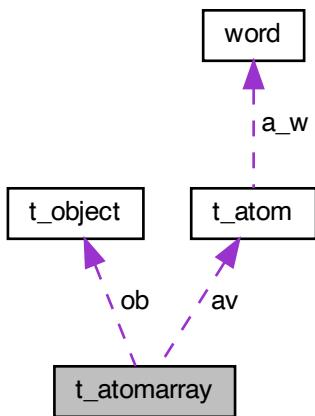
An atom is a typed datum.

## 39.2 t\_atomarray Struct Reference

The atomarray object.

```
#include <ext_atomarray.h>
```

Collaboration diagram for t\_atomarray:



### 39.2.1 Detailed Description

The atomarray object.

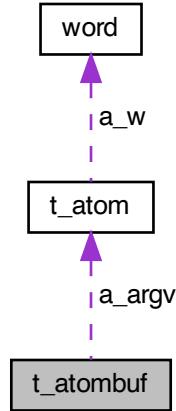
This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

## 39.3 t\_atombuf Struct Reference

The atombuf struct provides a way to pass a collection of atoms.

```
#include <ext_atombuf.h>
```

Collaboration diagram for t\_atombuf:



## Data Fields

- long `a_argc`  
*the number of atoms*
- `t_atom a_argv [1]`  
*the first of the array of atoms*

### 39.3.1 Detailed Description

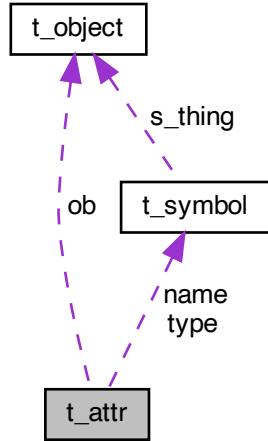
The atombuf struct provides a way to pass a collection of atoms.

## 39.4 t\_attr Struct Reference

Common attr struct.

```
#include <ext_obex.h>
```

Collaboration diagram for t\_attr:



### 39.4.1 Detailed Description

Common attr struct.

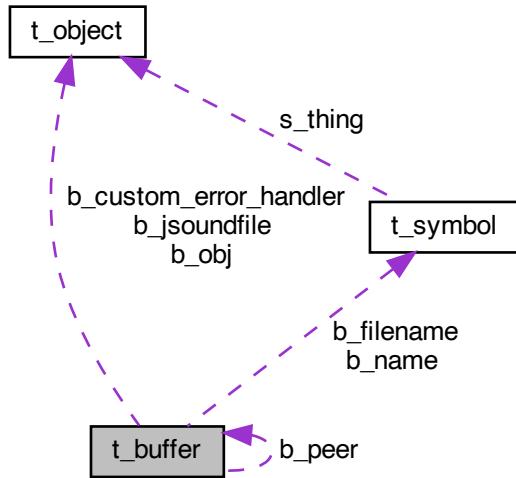
This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

## 39.5 t\_buffer Struct Reference

Data structure for the buffer~ object.

```
#include <buffer.h>
```

Collaboration diagram for t\_buffer:



## Data Fields

- `t_object b_obj`  
*doesn't have any signals so it doesn't need to be pxobject*
- `long b_valid`  
*flag is off during read replacement or editing operation*
- `float * b_samples`  
*stored with interleaved channels if multi-channel*
- `long b_frames`  
*number of sample frames (each one is sizeof(float) \* b\_nchans bytes)*
- `long b_nchans`  
*number of channels*
- `long b_size`  
*size of buffer in floats*
- `float b_sr`  
*sampling rate of the buffer*
- `float b_1oversr`  
*1 / sr*
- `float b_msrv`  
*sr \* .001*
- `float * b_memory`  
*pointer to where memory starts (initial padding for interp)*
- `t_symbol * b_name`  
*name of the buffer*

- long **b\_susloopstart**  
*looping info (from AIFF file) in samples*
- long **b\_susloopend**  
*looping info (from AIFF file) in samples*
- long **b\_reloopstart**  
*looping info (from AIFF file) in samples*
- long **b\_reloopend**  
*looping info (from AIFF file) in samples*
- long **b\_format**  
*'AIFF' or 'Sd2f'*
- **t\_symbol \* b\_filename**  
*last file read (not written) for info~*
- long **b\_olchnchans**  
*used for resizing window in case of # of channels change*
- long **b\_outputbytes**  
*number of bytes used for output sample (1-4)*
- long **b\_modtime**  
*last modified time ("dirty" method)*
- struct **\_buffer \* b\_peer**  
*objects that share this symbol (used as a link in the peers)*
- **t\_bool b\_owner**  
*b\_memory/b\_samples "owned" by this object*
- long **b\_outputfmt**  
*sample type (A\_LONG, A\_FLOAT, etc.)*
- **t\_int32\_atomic b\_inuse**  
*objects that use buffer should ATOMIC\_INCREMENT / ATOMIC\_DECREMENT this in their perform*
- void \* **b\_dspchain**  
*dspchain used for this instance*
- long **b\_padding**  
*amount of padding (number of samples) in b\_memory before b\_samples starts*
- long **b\_paddingchanged**  
*flag indicating that b\_padding has changed and needs to be allocated*
- **t\_object \* b\_jsoundfile**  
*internal instance for reading/writing FLAC format*
- **t\_systhread\_mutex b\_mutex**  
*mutex to use when locking and performing operations anywhere except perform method*
- long **b\_wasvalid**  
*internal flag used by replacement or editing operation*
- **t\_object \* b\_custom\_error\_handler**  
*used to return error numbers to a caller if this object is embedded inside of another object (e.g. playlist~)*
- **t\_clock \* b\_dirty\_clock**  
*used to return error numbers to a caller if this object is embedded inside of another object (e.g. playlist~)*
- **t\_qelem \* b\_dirty\_qelem**  
*used to move buffer dirty notifications to the main thread*
- **t\_bool b\_dirty\_done**  
*a buffer is not only dirty, but needs the 'done' message sent out its b\_doneout outlet*
- **t\_filepath b\_filepath**  
*path of last file read (not written)*

### 39.5.1 Detailed Description

Data structure for the buffer~ object.

Depreciated. Use [t\\_buffer\\_ref](#) and [t\\_buffer\\_obj](#) instead.

### 39.5.2 Field Documentation

#### 39.5.2.1 b\_dirty\_clock

```
t_clock* t_buffer::b_dirty_clock
```

used to return error numbers to a caller if this object is embedded inside of another object (e.g. playlist~)

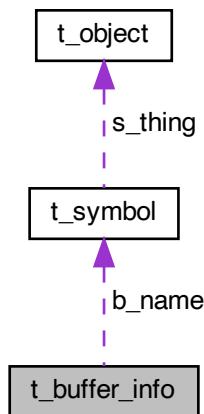
used to move buffer dirty notifications to the main thread

## 39.6 t\_buffer\_info Struct Reference

Common buffer~ data/metadata.

```
#include <ext_buffer.h>
```

Collaboration diagram for t\_buffer\_info:



## Data Fields

- `t_symbol * b_name`  
*name of the buffer*
- `float * b_samples`  
*stored with interleaved channels if multi-channel*
- `long b_frames`  
*number of sample frames (each one is sizeof(float) \* b\_nchans bytes)*
- `long b_nchans`  
*number of channels*
- `long b_size`  
*size of buffer in floats*
- `float b_sr`  
*sampling rate of the buffer*
- `long b_modtime`  
*last modified time ("dirty" method)*
- `long b_rfу [57]`  
*reserved for future use (total struct size is 64x4 = 256 bytes)*

### 39.6.1 Detailed Description

Common buffer~ data/metadata.

This info can be retrieved from a buffer~ using the `buffer_getinfo()` call.

## 39.7 t\_celldesc Struct Reference

A dataview cell description.

```
#include <jdataview.h>
```

### 39.7.1 Detailed Description

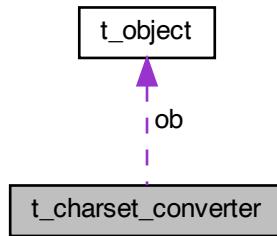
A dataview cell description.

## 39.8 t\_charset\_converter Struct Reference

The charset\_converter object.

```
#include <ext_charset.h>
```

Collaboration diagram for t\_charset\_converter:



### 39.8.1 Detailed Description

The charset\_converter object.

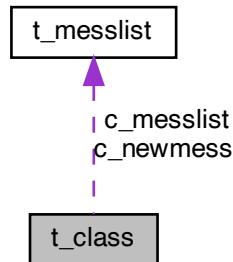
This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

## 39.9 t\_class Struct Reference

The data structure for a Max class.

```
#include <ext_mess.h>
```

Collaboration diagram for t\_class:



## Data Fields

- struct symbol \* `c_sym`  
*symbol giving name of class*
- struct symbol \* `c_filename`  
*name of file associated with this class*

### 39.9.1 Detailed Description

The data structure for a Max class.

This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

## 39.10 `t_datetime` Struct Reference

The Systime data structure.

```
#include <ext_systime.h>
```

## Data Fields

- `t_uint32 year`  
*year*
- `t_uint32 month`  
*month, in range 1 through 12*
- `t_uint32 day`  
*day, in range 1 through 31*
- `t_uint32 hour`  
*hour*
- `t_uint32 minute`  
*minute*
- `t_uint32 second`  
*second*
- `t_uint32 millisecond`  
*(reserved for future use)*

### 39.10.1 Detailed Description

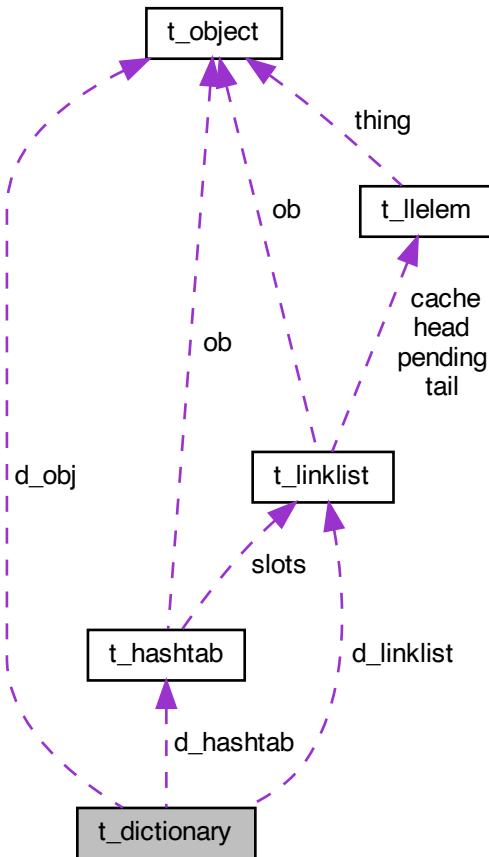
The Systime data structure.

## 39.11 t\_dictionary Struct Reference

The dictionary object.

```
#include <ext_dictionary.h>
```

Collaboration diagram for t\_dictionary:



### 39.11.1 Detailed Description

The dictionary object.

This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

See also

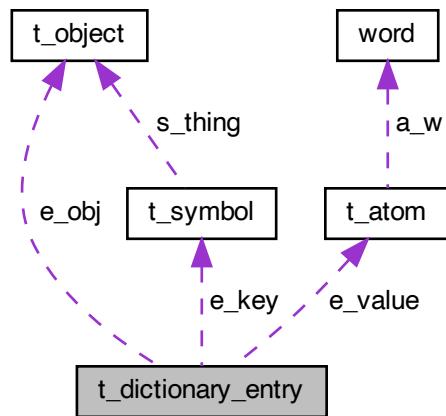
[t\\_dictionary](#)

## 39.12 t\_dictionary\_entry Struct Reference

A dictionary entry.

```
#include <ext_dictionary.h>
```

Collaboration diagram for t\_dictionary\_entry:



### 39.12.1 Detailed Description

A dictionary entry.

This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

See also

[t\\_dictionary](#)

## 39.13 t\_ex\_ex Struct Reference

`ex_ex`.

```
#include <ext_expr.h>
```

## Data Fields

- 
- union {  
} ex\_cont
- content
- long ex\_type  
*type of the node*

### 39.13.1 Detailed Description

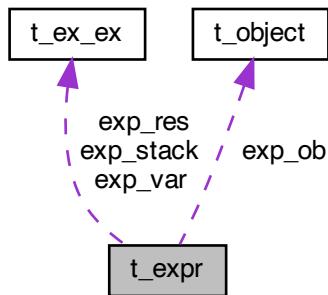
ex\_ex.

## 39.14 t\_expr Struct Reference

Struct for an instance of expr.

```
#include <ext_expr.h>
```

Collaboration diagram for t\_expr:



## Data Fields

- t\_ex\_ex exp\_res  
*the result of last evaluation*

### 39.14.1 Detailed Description

Struct for an instance of expr.

## 39.15 t\_fileinfo Struct Reference

Information about a file.

```
#include <ext_path.h>
```

### Data Fields

- [t\\_fourcc type](#)  
*type (four-char-code)*
- [t\\_fourcc creator](#)  
*Mac-only creator (four-char-code)*
- [t\\_uint32 unused](#)  
*this was date but it wasn't populated and it wasn't used*
- [t\\_int32 flags](#)  
*One of the values defined in [e\\_max\\_fileinfo\\_flags](#).*

### 39.15.1 Detailed Description

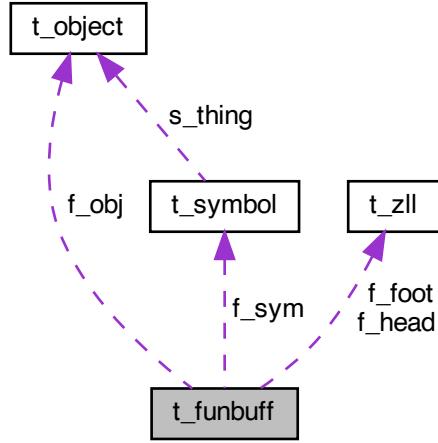
Information about a file.

## 39.16 t\_funbuff Struct Reference

The structure of a funbuff object.

```
#include <ext_maxtypes.h>
```

Collaboration diagram for t\_funbuff:



## Data Fields

- `t_zll f_head`  
*head of double linked list of function elements*
- `t_zll * f_foot`  
*foot in the door pointer for list*
- long `f_gotoDelta`  
*used by goto and next*
- long `f_selectX`  
*selected region start*
- long `f_selectW`  
*selected region width*
- `t_symbol * f_sym`  
*filename*
- long `f_y`  
*y-value from inlet*
- char `f_yvalid`  
*flag that y has been set since x has*
- char `f_embed`  
*flag for embedding funbuff values in patcher*

### 39.16.1 Detailed Description

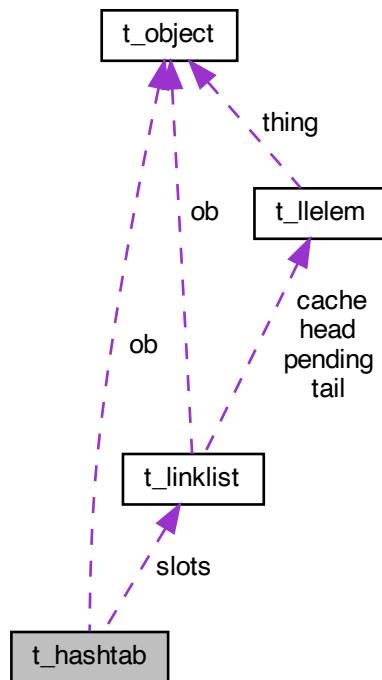
The structure of a funbuff object.

## 39.17 t\_hashtab Struct Reference

The hashtab object.

```
#include <ext_hashtab.h>
```

Collaboration diagram for t\_hashtab:



### 39.17.1 Detailed Description

The hashtab object.

This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

See also

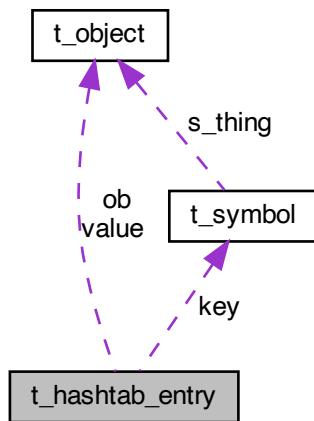
[t\\_hashtab](#)

## 39.18 t\_hashtab\_entry Struct Reference

A hashtab entry.

```
#include <ext_hashtab.h>
```

Collaboration diagram for t\_hashtab\_entry:



### 39.18.1 Detailed Description

A hashtab entry.

This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

See also

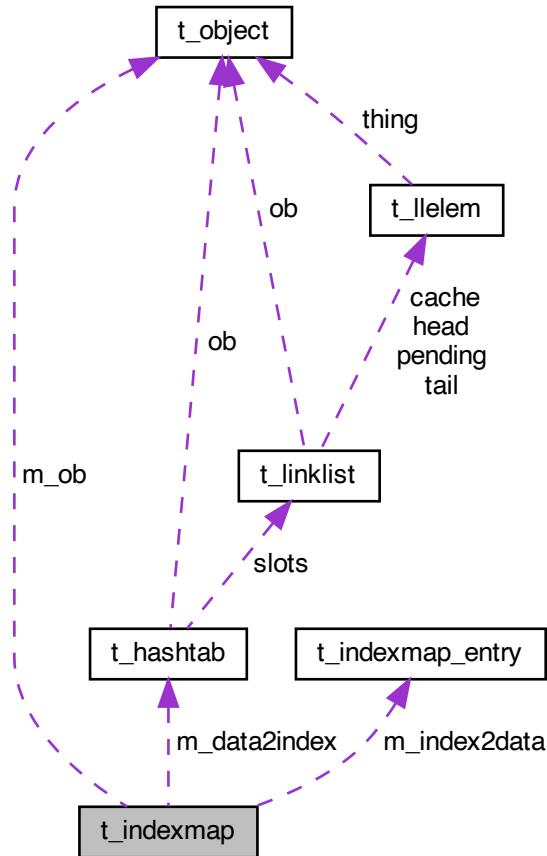
[t\\_hashtab](#)

## 39.19 t\_indexmap Struct Reference

An indexmap object.

```
#include <indexmap.h>
```

Collaboration diagram for t\_indexmap:



### 39.19.1 Detailed Description

An indexmap object.

This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

See also

[t\\_indexmap\\_entry](#)

## 39.20 t\_indexmap\_entry Struct Reference

An indexmap element.

```
#include <indexmap.h>
```

### 39.20.1 Detailed Description

An indexmap element.

This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

See also

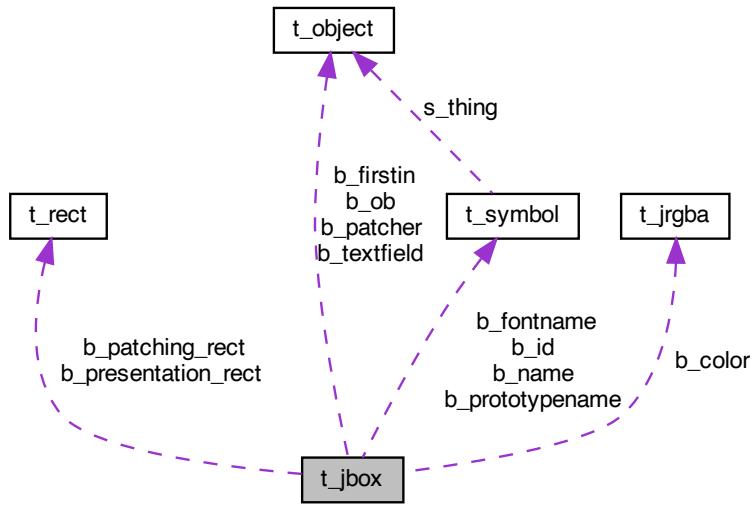
[t\\_indexmap](#)

## 39.21 t\_jbox Struct Reference

The [t\\_jbox](#) struct provides the header for a Max user-interface object.

```
#include <jpatcher_api.h>
```

Collaboration diagram for [t\\_jbox](#):



### 39.21.1 Detailed Description

The [t\\_jbox](#) struct provides the header for a Max user-interface object.

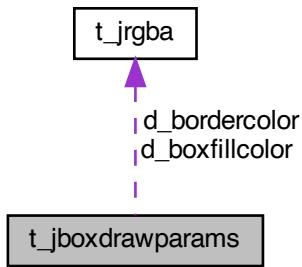
This struct should be considered opaque and is subject to change without notice. Do not access its members directly in any code.

## 39.22 t\_jboxdrawparams Struct Reference

The [t\\_jboxdrawparams](#) structure.

```
#include <jpatcher_api.h>
```

Collaboration diagram for t\_jboxdrawparams:



### 39.22.1 Detailed Description

The [t\\_jboxdrawparams](#) structure.

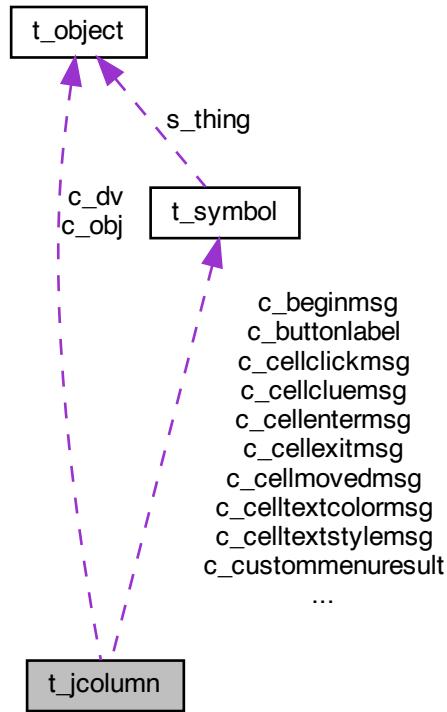
This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

## 39.23 t\_jcolumn Struct Reference

A dataview column.

```
#include <jdataview.h>
```

Collaboration diagram for t\_jcolumn:



## Data Fields

- `t_symbol * c_name`  
*column name (hash)*
- `t_object * c_dv`  
*parent dataview*
- int `c_id`  
*id in DataViewComponent*
- long `c_width`  
*column width in pixels*
- long `c_maxwidth`  
*max column width*
- long `c_minwidth`  
*min column width*
- char `c_autosize`  
*determine width of text column automatically (true/false)*
- char `c_alignment`

- display of text, left, right, center*
- **t\_symbol \* c\_font**  
*name of font*
- long **c\_fontsiz**  
*font size (points?)*
- **t\_symbol \* c\_label**  
*heading of column*
- char **c\_separator**  
*separator mode*
- char **c\_button**  
*column has a button (true/false)*
- **t\_symbol \* c\_buttonlabel**  
*text in a button*
- **t\_symbol \* c\_customsort**  
*message sent to sort this column – if none, default sorting is used based on value c\_numeric*
- char **c\_overridesort**  
*if true only the sortdata method is called, not the sort method (true/false)*
- **t\_symbol \* c\_custompaint**  
*send this msg name to client to paint this column*
- **t\_symbol \* c\_valuemsg**  
*message sent when a component mode cell's value changes*
- **t\_symbol \* c\_beginmsg**  
*message sent when a component mode cell's value is about to start changing*
- **t\_symbol \* c\_endmsg**  
*message sent when a component mode cell's value is finished changing*
- **t\_symbol \* c\_rowcomponentmsg**  
*message sent to determine what kind of component should be created for each cell in a column*
- **t\_symbol \* c\_custommenuset**  
*message to set a menu (for a readonly or custompaint column)*
- **t\_symbol \* c\_custommenuresult**  
*message sent when an item is chosen from a custom menu*
- char **c\_editable**  
*can you edit the data in a cell in this column*
- char **c\_selectable**  
*can select the data in a cell in this column (possibly without being able to edit)*
- char **c\_multiselectable**  
*can you select more than one cell in this column*
- char **c\_sortable**  
*can you click on a column heading to sort the data*
- long **c\_initiallysorted**  
*if this is set to JCOLUMN\_INITIALYSORTED\_FORWARD the column is displayed with the sort triangle*
- long **c\_maxtextlen**  
*maximum text length: this is used to allocate a buffer to pass to gettext (but there is also a constant)*
- long **c\_sortdirection**  
*0 for ascending, 1 for descending*
- long **c\_component**  
*enum of components (check box etc.)*

- char `c_canselect`  
*can select entire column*
- char `c_cancut`  
*can cut/clear entire column*
- char `c_cancopy`  
*can copy entire column*
- char `c_cancutcells`  
*can cut a single cell (assumes "editable" or "selectable") (probably won't be implemented)*
- char `c_cancopycells`  
*can copy a single cell*
- char `c_canpastecells`  
*can paste into a single cell*
- char `c_hideable`  
*can the column be hidden*
- char `c_hidden`  
*is the column hidden (set/get)*
- char `c_numeric`  
*is the data numeric (i.e., is getcellvalue implemented)*
- char `c_draggable`  
*can drag the column to rearrange it*
- char `c_casesensitive`  
*use case sensitive sorting (applies only to default text sorting)*
- char `c_showinfo`  
*show info button for cell clue on mouse over*
- void \* `c_reference`  
*reference for the use of the client*
- double `c_indentspacing`  
*amount of space (in pixels) for one indent level*
- `t_symbol * c_insertbefore`  
*name of column before which this one should have been inserted (used only once)*
- `t_symbol * c_cellcluemsg`  
*message to send requesting clue text for a cell*
- `t_symbol * c_celltextcolormsg`  
*message to get the cell's text color*
- `t_symbol * c_celltextstylemsg`  
*message to get the cell's style and alignment*
- `t_symbol * c_cellentermsg`  
*message for cell enter*
- `t_symbol * c_cellexitmsg`  
*message for cell exit*
- `t_symbol * c_cellmovedmsg`  
*message for cell mouse move*
- `t_symbol * c_cellclickmsg`  
*message for custom cell click action*

### 39.23.1 Detailed Description

A dataview column.

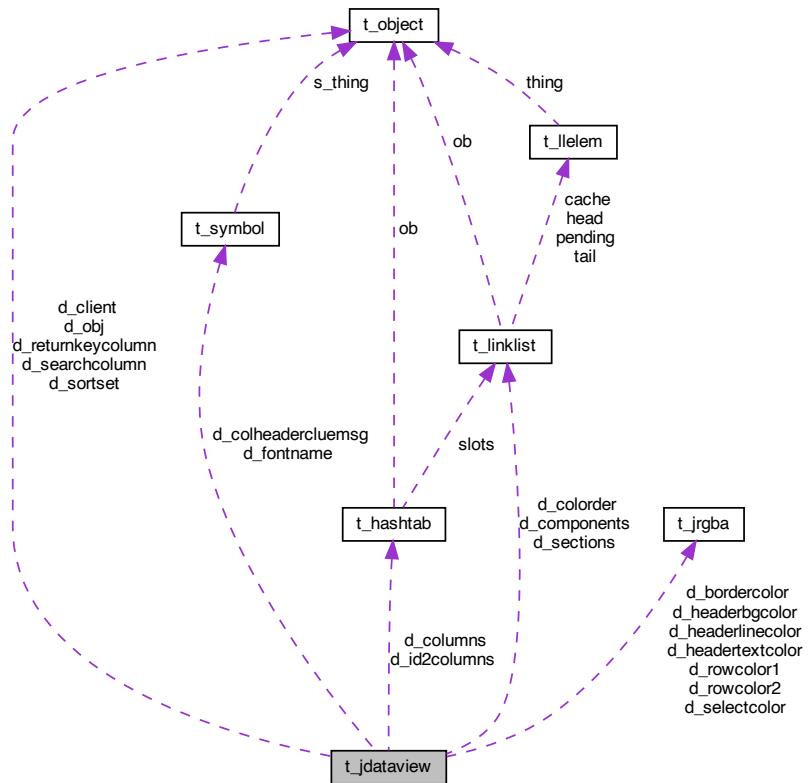
Columns for a given dataview are stored in a [t\\_hashtab](#) and accessed by name.

## 39.24 t\_jdatatable Struct Reference

The datatable object.

```
#include <jdatatable.h>
```

Collaboration diagram for t\_jdatatable:



## Data Fields

- [t\\_linklist \\* d\\_components](#)  
*list of DataViewComponents showing this dataview*
- [t\\_object \\* d\\_client](#)

- object that will be sent messages to get data to display
- `t_hashtab * d_columns`  
*columns – point to `t_jcolumn` objects*
- `t_hashtab * d_id2columns`  
*columns from column IDs*
- `t_linklist * d_colororder`  
*current order of columns*
- `void * d_rowmap_obsolete`  
*no longer used*
- `long d_numcols`  
*number of columns*
- `double d_rowheight`  
*fixed height of a row in pixels*
- `char d_autoheight`  
*height determined by font*
- `char d_hierarchical`  
*does it allow hierarchical disclosure (true / false) – not implemented yet*
- `t_jrgba d_rowcolor1`  
*odd row color (striped)*
- `t_jrgba d_rowcolor2`  
*even row color*
- `t_jrgba d_selectcolor`  
*color when rows are selected*
- `t_jrgba d_bordercolor`  
*border color*
- `char d_bordercolorset`  
*was border color set? if not, use JUCE default*
- `char d_canselectmultiple`  
*multiple rows are selectable*
- `char d_cancopy`  
*copy enabled*
- `char d_cancut`  
*cut / clear enabled*
- `char d_canpaste`  
*paste enabled*
- `char d_canrearrangerows`  
*rows can be dragged to rearrange – may not be implemented yet*
- `char d_canrearrangecolumns`  
*columns can be dragged to rearrange*
- `long d_viscount`  
*number of visible views of this dataview*
- `long d_inset`  
*inset for table inside containing component in pixels*
- `char d_autosizeright`  
*right side autosizes when top-level component changes*
- `char d_autosizebottom`  
*bottom autosizes when top-level component changes*

- char **d\_dragenabled**  
*enabled for dragging (as in drag and drop)*
- **t\_symbol \* d\_fontname**  
*font name*
- double **d\_fontsize**  
*font size*
- **t\_symbol \* d\_colheadercluemsg**  
*message to send requesting clue text for the column headers*
- char **d\_autosizerightcolumn**  
*right column should stretch to remaining width of the dataview, regardless of column width*
- char **d\_customselectcolor**  
*send getcellcolor message to draw selected cell, don't use select color*
- void \* **d\_qelem**  
*defer updating*
- long **d\_top\_inset**  
*vertical inset for row background (default 0)*
- long **d\_bottom\_inset**  
*vertical inset for row background (default 0)*
- long **d\_borderthickness**  
*border line thickness default 0 for no border*
- char **d\_keyfocusable**  
*notify component to grab some keys*
- char **d\_enabledeletekey**  
*delete key will delete selected rows*
- char **d\_usegradient**  
*color rows with gradient between rowcolor1 (top) and rowcolor2 (bottom)*
- char **d\_inchange**  
*in change flag for inspector end-change protection system*
- char **d\_horizscrollvisible**  
*is horizontal scroll bar visible*
- char **d\_vertscrollvisible**  
*is vertical scroll bar visible*
- char **d\_scrollvisset**  
*has the scroll visibility ever been changed since the dv was created?*
- char **d\_overridefocus**  
*override default focus behavior where ListBox is focused when assigning focus to the dataview*
- char **d\_usesystemfont**  
*use system font (true by default)*
- **t\_object \* d\_searchcolumn**  
*column we ask for celltext in order to navigate the selection via the keyboard*
- **t\_object \* d\_returnkeycolumn**  
*column that is sent the return key when a given row is selected*
- void \* **d\_navcache**  
*sorted list of column strings for key navigation*
- char **d\_usecharheight**  
*use font specified in points rather than pixels (default is pixels)*
- **t\_linklist \* d\_sections**

- *list of sections*
- char [d\\_paintcellseparator](#)  
*should paint a line below a cell (grayish)*
- [t\\_object \\* d\\_sortset](#)  
*sort col saved when dv is invisible*

### 39.24.1 Detailed Description

The dataview object.

## 39.25 t\_jgraphics\_font\_extents Struct Reference

A structure for holding information related to how much space the rendering of a given font will use.

```
#include <jgraphics.h>
```

### Data Fields

- double [ascent](#)  
*The ascent.*
- double [descent](#)  
*The descent.*
- double [height](#)  
*The hieght.*
- double [max\\_x\\_advance](#)  
*Unused / Not valid.*
- double [max\\_y\\_advance](#)  
*Unused / Not valid.*

### 39.25.1 Detailed Description

A structure for holding information related to how much space the rendering of a given font will use.

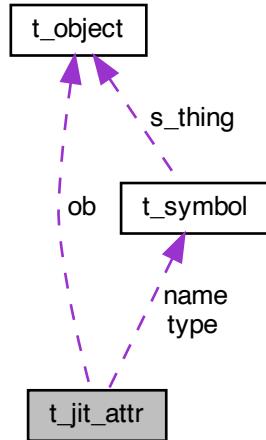
The units for these measurements is in pixels.

## 39.26 t\_jit\_attr Struct Reference

Common attribute struct.

```
#include <jit.common.h>
```

Collaboration diagram for t\_jit\_attr:



### Data Fields

- `t_jit_object ob`  
*common object header*
- `t_symbol * name`  
*attribute name*
- `t_symbol * type`  
*attribute type (char, long, float32, float64, symbol, atom, or obj)*
- `long flags`  
*flags for public/private get/set methods*
- `method get`  
*override default get method*
- `method set`  
*override default set method*
- `void * filterget`  
*filterobject for get method*
- `void * filterset`  
*filterobject for set method*
- `void * reserved`  
*for future use*

### 39.26.1 Detailed Description

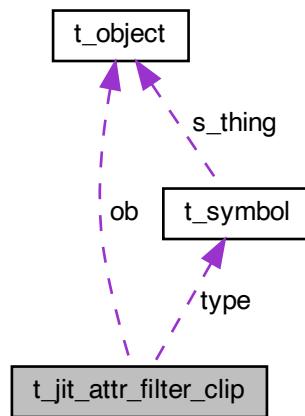
Common attribute struct.

Shared by all built in attribute classes.

## 39.27 t\_jit\_attr\_filter\_clip Struct Reference

[t\\_jit\\_attr\\_filter\\_clip](#) object struct.

Collaboration diagram for t\_jit\_attr\_filter\_clip:



### Data Fields

- `t_jit_object ob`  
*common object header*
- `t_symbol * type`  
*"type" attribute*
- `double scale`  
*scaling factor; "scale" attribute*
- `double min`  
*minimum value; "min" attribute*
- `double max`  
*maximum value; "max" attribute*
- `char usescale`  
*use scaling flag; "usescale" attribute*
- `char usemin`  
*clip to minimum flag; "usemin" attribute*
- `char usemax`  
*clip to maximum flag; "usemax" attribute*

### 39.27.1 Detailed Description

[t\\_jit\\_attr\\_filter\\_clip](#) object struct.

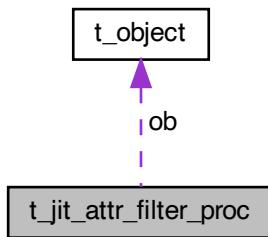
#### Warning

This struct should not be accessed directly, but is provided for reference when calling Jitter attribute functions.

## 39.28 t\_jit\_attr\_filter\_proc Struct Reference

[t\\_jit\\_attr\\_filter\\_proc](#) object struct.

Collaboration diagram for t\_jit\_attr\_filter\_proc:



### Data Fields

- [t\\_jit\\_object ob](#)  
*common object header*
- method [proc](#)  
*filter procedure*

### 39.28.1 Detailed Description

[t\\_jit\\_attr\\_filter\\_proc](#) object struct.

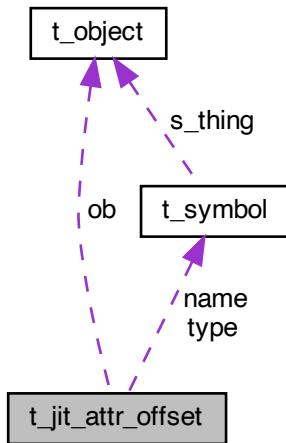
#### Warning

This struct should not be accessed directly, but is provided for reference when calling Jitter attribute functions.

## 39.29 t\_jit\_attr\_offset Struct Reference

[t\\_jit\\_attr\\_offset](#) object struct.

Collaboration diagram for t\_jit\_attr\_offset:



### Data Fields

- `t_jit_object ob`  
*common object header*
- `t_symbol * name`  
*attribute name*
- `t_symbol * type`  
*attribute type (char, long, float32, float64, symbol, atom, or obj)*
- `long flags`  
*flags for public/private get/set methods*
- method `get`  
*override default get method*
- method `set`  
*override default set method*
- `void * filterget`  
*filterobject for get method*
- `void * filterset`  
*filterobject for set method*
- `void * reserved`  
*for future use*
- `long offset`  
*byte offset to the attribute data*

### 39.29.1 Detailed Description

[t\\_jit\\_attr\\_offset](#) object struct.

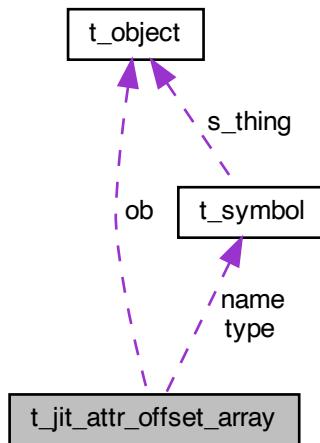
#### Warning

This struct should not be accessed directly, but is provided for reference. Attribute objects do not typically use attributes themselves to access members, but rather accessor methods—i.e. use `jit_object_method` in place of the `jit_attr_*` functions to access attribute state.

## 39.30 t\_jit\_attr\_offset\_array Struct Reference

[t\\_jit\\_attr\\_offset\\_array](#) object struct.

Collaboration diagram for `t_jit_attr_offset_array`:



## Data Fields

- [t\\_jit\\_object ob](#)  
*common object header*
- [t\\_symbol \\* name](#)  
*attribute name*
- [t\\_symbol \\* type](#)  
*attribute type (char, long, float32, float64, symbol, atom, or obj)*
- long [flags](#)  
*flags for public/private get/set methods*

- method `get`  
*override default get method*
- method `set`  
*override default set method*
- void \* `filterget`  
*filterobject for get method*
- void \* `filterset`  
*filterobject for set method*
- void \* `reserved`  
*for future use*
- long `offset`  
*byte offset to the attribute data*
- long `size`  
*maximum size*
- long `offsetcount`  
*byte offset to the attribute count*

### 39.30.1 Detailed Description

`t_jit_attr_offset_array` object struct.

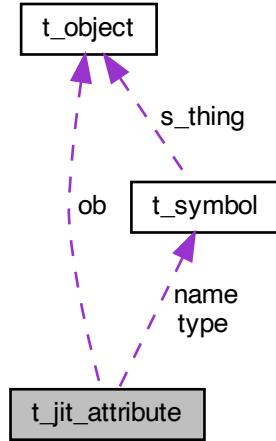
#### Warning

This struct should not be accessed directly, but is provided for reference. Attribute objects do not typically use attributes themselves to access members, but rather accessor methods—i.e. use `jit_object_method` in place of the `jit_attr_*` functions to access attribute state.

## 39.31 `t_jit_attribute` Struct Reference

`t_jit_attribute` object struct.

Collaboration diagram for t\_jit\_attribute:



## Data Fields

- `t_jit_object ob`  
*common object header*
- `t_symbol * name`  
*attribute name*
- `t_symbol * type`  
*attribute type (char, long, float32, float64, symbol, atom, or obj)*
- `long flags`  
*flags for public/private get/set methods*
- method `get`  
*override default get method*
- method `set`  
*override default set method*
- `void * filterget`  
*filterobject for get method*
- `void * filterset`  
*filterobject for set method*
- `void * reserved`  
*for future use*
- `void * data`  
*internally stored data*
- `long size`  
*data size*

### 39.31.1 Detailed Description

[t\\_jit\\_attribute](#) object struct.

#### Warning

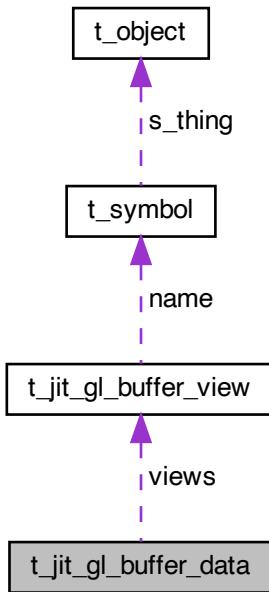
This struct should not be accessed directly, but is provided for reference. Attribute objects do not typically use attributes themselves to access members, but rather accessor methods—i.e. use `jit_object_method` in place of the `jit_attr_*` functions to access attribute state.

## 39.32 t\_jit\_gl\_buffer\_data Struct Reference

Represents data to be stored in a buffer.

```
#include <jit/gl/draw.h>
```

Collaboration diagram for `t_jit_gl_buffer_data`:



### 39.32.1 Detailed Description

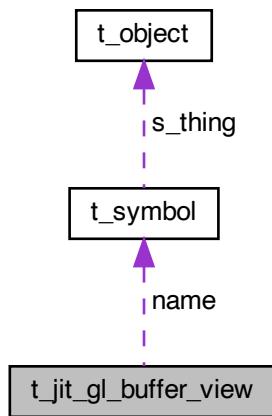
Represents data to be stored in a buffer.

### 39.33 t\_jit\_gl\_buffer\_view Struct Reference

Data structures and functions for OpenGL drawing.

```
#include <jit.gl.draw.h>
```

Collaboration diagram for t\_jit\_gl\_buffer\_view:



#### 39.33.1 Detailed Description

Data structures and functions for OpenGL drawing.

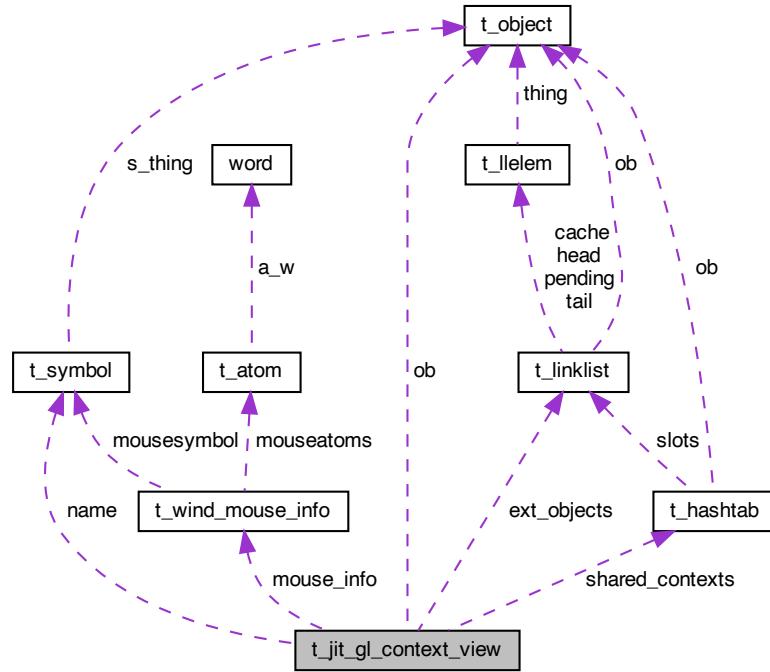
Represents a view into a buffer, for use with glVertexAttribPointer etc.

### 39.34 t\_jit\_gl\_context\_view Struct Reference

`t_jit_gl_context_view` object struct.

```
#include <jit.gl.context.view.h>
```

Collaboration diagram for t\_jit\_gl\_context\_view:



## Data Fields

- **t\_object ob**  
*jitter object*
- long **rebuild**  
*rebuild flag*
- **t\_jit\_gl\_context context**  
*OpenGL context.*
- **t\_wind\_mouse\_info mouse\_info**  
*data for mouse events*
- **t\_wind\_mousewheel\_info mousewheel\_info**  
*data for mouse wheel events*
- **t\_wind\_key\_info key\_info**  
*data for key events*
- long **canrebuild**  
*flag for whether the context can rebuild or not*
- long **doublebuffer**  
*double buffer flag*
- long **depthbuffer**  
*depth buffer flag*

- long **stereo**  
*active stereo flag*
- t\_jit\_rect **frame**  
*frame of context*
- long **fsaa**  
*FSAA flag.*
- long **sync**  
*V-sync flag.*
- long **shared**  
*Shader context flag.*
- t\_hashtab \* **shared\_contexts**  
*Hashtab of shared context names.*
- long **idlemouse**  
*Idlemouse flag (events on mouse move)*
- long **mousewheel**  
*mosuewheel flag (events on mouse wheel)*
- void \* **target**  
*target object we're controlled by*
- long **targettype**  
*target type we're controlled by*
- t\_symbol \* **name**  
*name of the view*
- long **reshaping**  
*flag for breaking cycles on reshape notification*
- long **ownerreshape**  
*flag for if the owner handles reshaping the context*
- long **freeing**  
*in the process of freeing flag*
- long **creating**  
*in the process of creating flag*
- long **destroying**  
*in the process of destroying flag*
- float **scalefactor**  
*scaling factor when drawing to retina display*
- long **allow\_hi\_res**  
*allows for high resolution drawing when available*
- void \* **nativewinhandle**  
*patcher native window handle, for offscreen contexts*

### 39.34.1 Detailed Description

t\_jit\_gl\_context\_view object struct.

Manages an OpenGL context within a rectangle. Objects that use a t\_jit\_gl\_context\_view to manage an OpenGL context should attach themselves to the object for its lifetime and implement an "update" method in order to handle modifications to the t\_jit\_gl\_context\_view that may require a rebuild or further response within the embedding object.

## 39.35 t\_jit\_gl\_drawinfo Struct Reference

[t\\_jit\\_gl\\_drawinfo](#) struct used for tasks such as multi texture unit binding.

```
#include <jit.gl.drawinfo.h>
```

### Data Fields

- `t_jit_gl_context ctx`  
*current t\_jit\_gl\_context*
- `void * ob3d`  
*object's t\_jit\_ob3d pointer*
- `void * rfu [6]`  
*reserved for future use*

### 39.35.1 Detailed Description

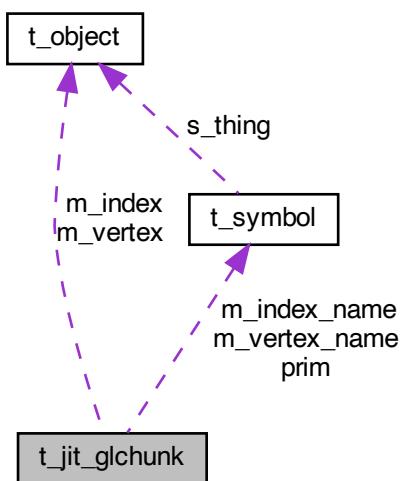
[t\\_jit\\_gl\\_drawinfo](#) struct used for tasks such as multi texture unit binding.

## 39.36 t\_jit\_glchunk Struct Reference

[t\\_jit\\_glchunk](#) is a public structure to store one gl-command's-worth of data, in a format which can be passed easily to glDrawRangeElements, and matrixoutput.

```
#include <jit.gl.chunk.h>
```

Collaboration diagram for t\_jit\_glchunk:



## Data Fields

- `t_symbol * prim`  
*drawing primitive. "tri\_strip", "tri", "quads", "quad\_grid", etc.*
- `t jit_object * m_vertex`  
*vertex matrix containing xyzst... data*
- `t_symbol * m_vertex_name`  
*vertex matrix name*
- `t jit_object * m_index`  
*optional 1d matrix of vertex indices to use with drawing primitive*
- `t_symbol * m_index_name`  
*index matrix name*
- `unsigned long m_flags`  
*chunk flags to ignore texture, normal, color, or edge planes when drawing*
- `void * next_chunk`  
*pointer to next chunk for drawing a list of chunks together*

### 39.36.1 Detailed Description

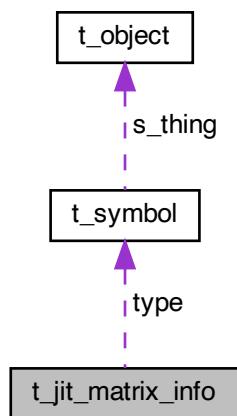
`t jit_glchunk` is a public structure to store one gl-command's-worth of data, in a format which can be passed easily to `glDrawRangeElements`, and `matrixoutput`.

## 39.37 `t jit_matrix_info` Struct Reference

Matrix information struct.

```
#include <jit.common.h>
```

Collaboration diagram for `t jit_matrix_info`:



## Data Fields

- long [size](#)  
*in bytes (0xFFFFFFFF=UNKNOWN)*
- [t\\_symbol \\*type](#)  
*primitive type (char, long, float32, or float64)*
- long [flags](#)  
*flags to specify data reference, handle, or tightly packed*
- long [dimcount](#)  
*number of dimensions*
- long [dim \[JIT\\_MATRIX\\_MAX\\_DIMCOUNT\]](#)  
*dimension sizes*
- long [dimstride \[JIT\\_MATRIX\\_MAX\\_DIMCOUNT\]](#)  
*stride across dimensions in bytes*
- long [planeCount](#)  
*number of planes*

### 39.37.1 Detailed Description

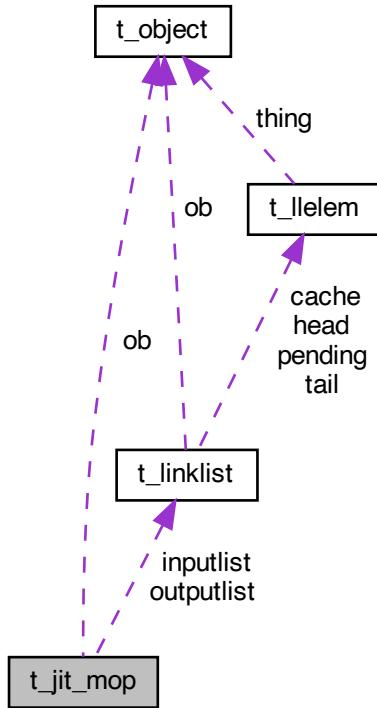
Matrix information struct.

Used to get/set multiple matrix attributes at once.

## 39.38 t\_jit\_mop Struct Reference

[t\\_jit\\_mop](#) object struct.

Collaboration diagram for t\_jit\_mop:



## Data Fields

- `t_jit_object ob`  
*standard object header*
- `void * special`  
*special data pointer for use by wrappers of various kinds (e.g. max wrapper)*
- `long inputcount`  
*"inputcount" attribute*
- `long outputcount`  
*"inputcount" attribute*
- `t_jit_linklist * inputlist`  
*linked list of inputs, accessed via methods*
- `t_jit_linklist * outputlist`  
*linked list of outputs, accessed via methods*
- `char caninplace`  
*deprecated*
- `char adapt`  
*"adapt" attribute*
- `char outputmode`  
*"outputmode" attribute*

### 39.38.1 Detailed Description

[t\\_jit\\_mop](#) object struct.

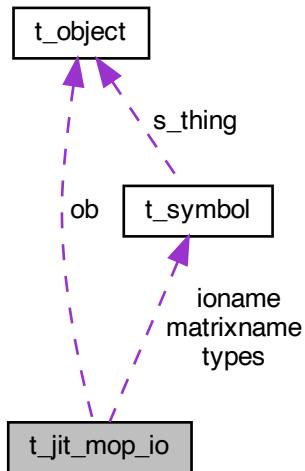
**Warning**

This struct should not be accessed directly, but is provided for reference when calling Jitter attribute functions.

## 39.39 t\_jit\_mop\_io Struct Reference

[t\\_jit\\_mop\\_io](#) object struct.

Collaboration diagram for [t\\_jit\\_mop\\_io](#):



## Data Fields

- **t\_jit\_object ob**  
*standard object header*
- **void \* special**  
*special data pointer for use by wrappers of various kinds (e.g. max wrapper)*
- **t\_symbol \* ioname**  
*"ioname" attribute*
- **t\_symbol \* matrixname**  
*"matrixname" attribute*
- **void \* matrix**

- internal matrix, accessed via methods (unused in class template MOP)*
- **t\_symbol \* types** [JIT\_MATRIX\_MAX\_TYPES]
  - "types" attribute
- long **mindim** [JIT\_MATRIX\_MAX\_DIMCOUNT]
  - "mindim" attribute
- long **maxdim** [JIT\_MATRIX\_MAX\_DIMCOUNT]
  - "maxdim" attribute
- long **typescount**
  - relevant to "types" attribute
- long **mindimcount**
  - "mindimcount" attribute
- long **maxdimcount**
  - "maxdimcount" attribute
- long **minplanecount**
  - "minplanecount" attribute
- long **maxplanecount**
  - "maxplanecount" attribute
- char **typelink**
  - "typelink" attribute
- char **dimlink**
  - "dimlink" attribute
- char **planelink**
  - "planelink" attribute
- method **ioproc**
  - I/O procedure, accessed via methods.

### 39.39.1 Detailed Description

**t\_jit\_mop\_io** object struct.

#### Warning

This struct should not be accessed directly, but is provided for reference when calling Jitter attribute functions.

## 39.40 t\_jit\_op\_info Struct Reference

Provides base pointer and stride for vector operator functions.

```
#include <jit.op.h>
```

### Data Fields

- void \* **p**
  - base pointer (coerced to appropriate type)*
- long **stride**
  - stride between elements (in type, not bytes)*

### 39.40.1 Detailed Description

Provides base pointer and stride for vector operator functions.

## 39.41 t\_jmatrix Struct Reference

An affine transformation (such as scale, shear, etc).

```
#include <jgraphics.h>
```

### Data Fields

- double **xx**  
*xx component*
- double **yx**  
*yx component*
- double **xy**  
*xy component*
- double **yy**  
*yy component*
- double **x0**  
*x translation*
- double **y0**  
*y translation*

### 39.41.1 Detailed Description

An affine transformation (such as scale, shear, etc).

## 39.42 t\_jrgb Struct Reference

A color composed of red, green, and blue components.

```
#include <jpatcher_api.h>
```

### Data Fields

- double **red**  
*Red component in the range [0.0, 1.0].*
- double **green**  
*Green component in the range [0.0, 1.0].*
- double **blue**  
*Blue component in the range [0.0, 1.0].*

### 39.42.1 Detailed Description

A color composed of red, green, and blue components.

Typically such a color is assumed to be completely opaque (with no transparency).

See also

[t\\_jrgba](#)

## 39.43 t\_rgba Struct Reference

A color composed of red, green, blue, and alpha components.

```
#include <jpatcher_api.h>
```

### Data Fields

- double [red](#)  
*Red component in the range [0.0, 1.0].*
- double [green](#)  
*Green component in the range [0.0, 1.0].*
- double [alpha](#)  
*Alpha (transparency) component in the range [0.0, 1.0].*

### 39.43.1 Detailed Description

A color composed of red, green, blue, and alpha components.

## 39.44 t\_line\_3d Struct Reference

Line or line segment in 3D space (GLfloat)

```
#include <jit/gl/common.h>
```

### Data Fields

- float [u](#) [3]  
*starting point*
- float [v](#) [3]  
*ending point*

### 39.44.1 Detailed Description

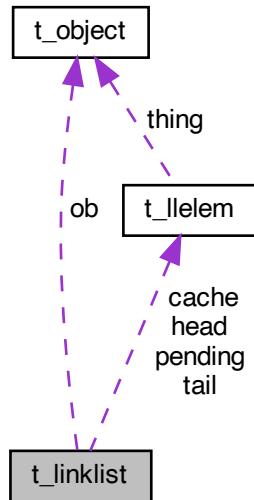
Line or line segment in 3D space (GLfloat)

## 39.45 t\_linklist Struct Reference

The linklist object.

```
#include <ext_linklist.h>
```

Collaboration diagram for t\_linklist:



### 39.45.1 Detailed Description

The linklist object.

This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

See also

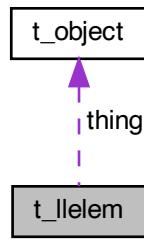
[t\\_llelem](#)

## 39.46 t\_ll elem Struct Reference

A linklist element.

```
#include <ext_linklist.h>
```

Collaboration diagram for t\_ll elem:



### 39.46.1 Detailed Description

A linklist element.

This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

See also

[t\\_linklist](#)

## 39.47 t\_matrix\_conv\_info Struct Reference

Matrix conversion struct.

```
#include <jit.common.h>
```

## Data Fields

- long `flags`  
*flags for whether or not to use interpolation, or source/destination dimensions*
- long `planemap` [JIT\_MATRIX\_MAX\_PLANECOUNT]  
*plane mapping*
- long `srcdimstart` [JIT\_MATRIX\_MAX\_DIMCOUNT]  
*source dimension start*
- long `srcdimend` [JIT\_MATRIX\_MAX\_DIMCOUNT]  
*source dimension end*
- long `dstdimstart` [JIT\_MATRIX\_MAX\_DIMCOUNT]  
*destination dimension start*
- long `dstdimend` [JIT\_MATRIX\_MAX\_DIMCOUNT]  
*destination dimension end*

### 39.47.1 Detailed Description

Matrix conversion struct.

Used to copy data from one matrix to another with special characteristics.

## 39.48 t\_messlist Struct Reference

A list of symbols and their corresponding methods, complete with typechecking information.

```
#include <ext_mess.h>
```

## Data Fields

- struct symbol \* `m_sym`  
*Name of the message.*
- method `m_fun`  
*Method associated with the message.*
- char `m_type` [MSG\_MAXARG+1]  
*Argument type information.*

### 39.48.1 Detailed Description

A list of symbols and their corresponding methods, complete with typechecking information.

## 39.49 t\_object Struct Reference

The structure for the head of any object which wants to have inlets or outlets, or support attributes.

```
#include <ext_mess.h>
```

### Data Fields

- struct messlist \* [o\\_messlist](#)  
*list of messages and methods. The -1 entry of the message list of an object contains a pointer to its [t\\_class](#) entry.*
- [t\\_ptr\\_int o\\_magic](#)  
*magic number*
- [t\\_inlet \\* o\\_inlet](#)  
*list of inlets*
- [t\\_outlet \\* o\\_outlet](#)  
*list of outlets*

### 39.49.1 Detailed Description

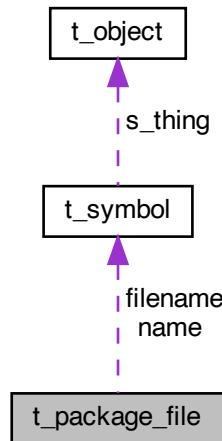
The structure for the head of any object which wants to have inlets or outlets, or support attributes.

## 39.50 t\_package\_file Struct Reference

A container for a path/filename pair, together with additional meta fields for special cases.

```
#include <ext_packages.h>
```

Collaboration diagram for t\_package\_file:



### 39.50.1 Detailed Description

A container for a path/filename pair, together with additional meta fields for special cases.

## 39.51 t\_path Struct Reference

The path data structure.

```
#include <ext_path.h>
```

### 39.51.1 Detailed Description

The path data structure.

This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

## 39.52 t\_pathlink Struct Reference

The pathlink data structure.

```
#include <ext_path.h>
```

### 39.52.1 Detailed Description

The pathlink data structure.

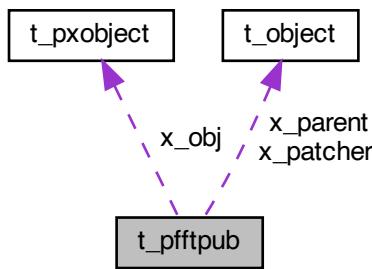
This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

## 39.53 t\_pfftpub Struct Reference

Public FFT Patcher struct.

```
#include <r_pfft.h>
```

Collaboration diagram for t\_pfftpub:



### Data Fields

- `t_object * x_parent`  
*parent patcher*
- `t_object * x_patch`  
*patcher loaded*
- `struct _dspchain * x_chain`  
*dsp chain within pfft*
- long `x_fftsize`  
*fft frame size*
- long `x_ffthop`  
*hop between fft frames*
- long `x_ftoffset`  
*n samples offset before fft is started*
- long `x_ftindex`  
*current index into fft frame*
- short `x_fullspect`  
*process half-spectrum (0) or full mirrored spectrum (1)?*

### 39.53.1 Detailed Description

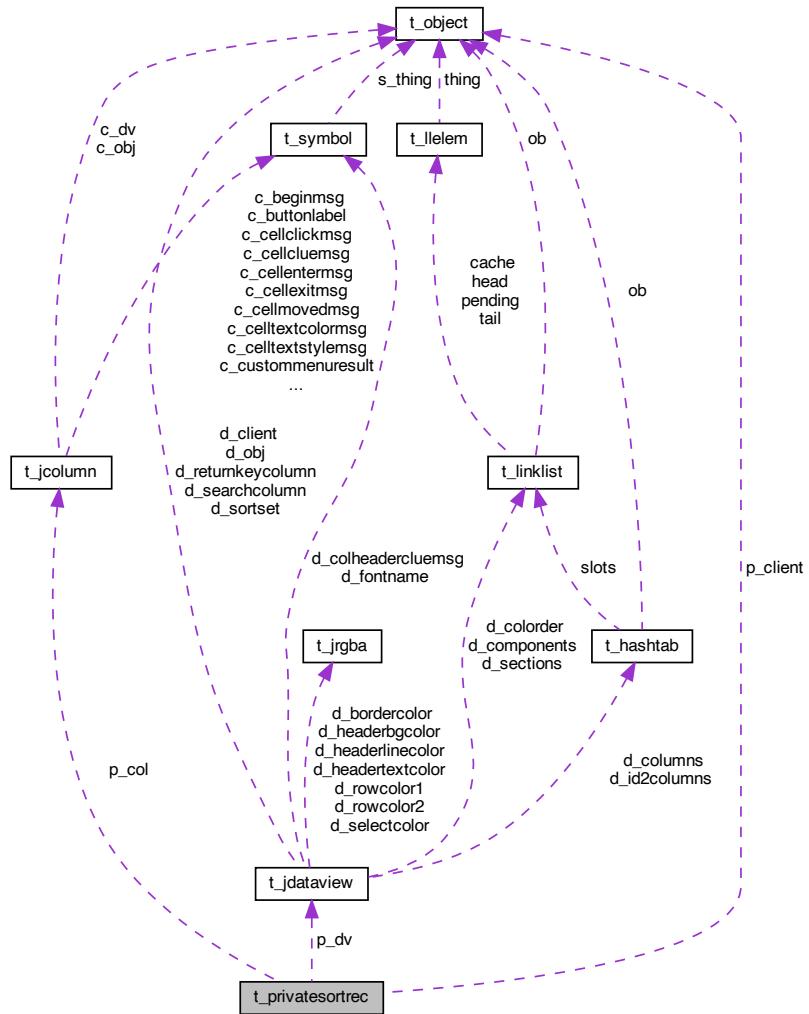
Public FFT Patcher struct.

## 39.54 t\_privatesortrec Struct Reference

used to pass data to a client sort function

```
#include <jdataview.h>
```

Collaboration diagram for t\_privatesortrec:



### Data Fields

- [t\\_jcolumn \\* p\\_col](#)  
*column object to sort*
- char [p\\_fwd](#)

- *1 if sorting "forwards"*
- [`t\_object \* p\_client`](#)  
*pointer to the client object*
- [`t\_jdataview \* p\_dv`](#)  
*pointer to the dataview*

### 39.54.1 Detailed Description

used to pass data to a client sort function

## 39.55 `t_pt` Struct Reference

Coordinates for specifying a point.

```
#include <jpatcher_api.h>
```

### Data Fields

- `double x`  
*The horizontal coordinate.*
- `double y`  
*The vertical coordinate.*

### 39.55.1 Detailed Description

Coordinates for specifying a point.

See also

[`t\_rect`](#)  
[`t\_size`](#)

## 39.56 `t_pxdata` Struct Reference

Common struct for MSP objects.

```
#include <z_dsp.h>
```

## Data Fields

- long `z_disabled`  
*set to non-zero if this object is muted (using the pcontrol or mute~ objects)*
  - short `z_count`  
*the number of signal inlets*
  - short `z_misc`  
*flags (bitmask) determining object behaviour, such as `Z_NO_INPLACE`, `Z_PUT_FIRST`, or `Z_PUT_LAST`*

### **39.56.1 Detailed Description**

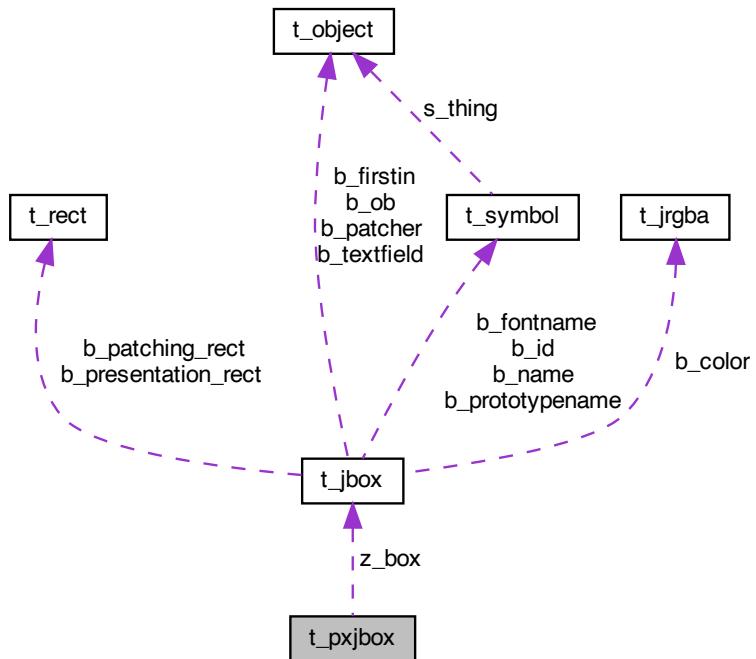
## Common struct for MSP objects.

## 39.57 t\_pxjbox Struct Reference

Header for any ui signal processing object.

```
#include <z_dsp.h>
```

## Collaboration diagram for t\_pxjbox:



## Data Fields

- [t\\_pxjbox](#) [z\\_box](#)  
*The box struct used by all ui objects.*
- long [z\\_disabled](#)  
*set to non-zero if this object is muted (using the pcontrol or mute~ objects)*
- short [z\\_count](#)  
*the number of signal inlets*
- short [z\\_misc](#)  
*flags (bitmask) determining object behaviour, such as [Z\\_NO\\_INPLACE](#), [Z\\_PUT\\_FIRST](#), or [Z\\_PUT\\_LAST](#)*

### 39.57.1 Detailed Description

Header for any ui signal processing object.

For non-ui objects use [t\\_pxobject](#).

## 39.58 t\_pxobject Struct Reference

Header for any non-ui signal processing object.

```
#include <z_dsp.h>
```

## Data Fields

- struct object [z\\_ob](#)  
*The standard [t\\_object](#) struct.*
- long [z\\_disabled](#)  
*set to non-zero if this object is muted (using the pcontrol or mute~ objects)*
- short [z\\_count](#)  
*the number of signal inlets*
- short [z\\_misc](#)  
*flags (bitmask) determining object behaviour, such as [Z\\_NO\\_INPLACE](#), [Z\\_PUT\\_FIRST](#), or [Z\\_PUT\\_LAST](#)*

### 39.58.1 Detailed Description

Header for any non-ui signal processing object.

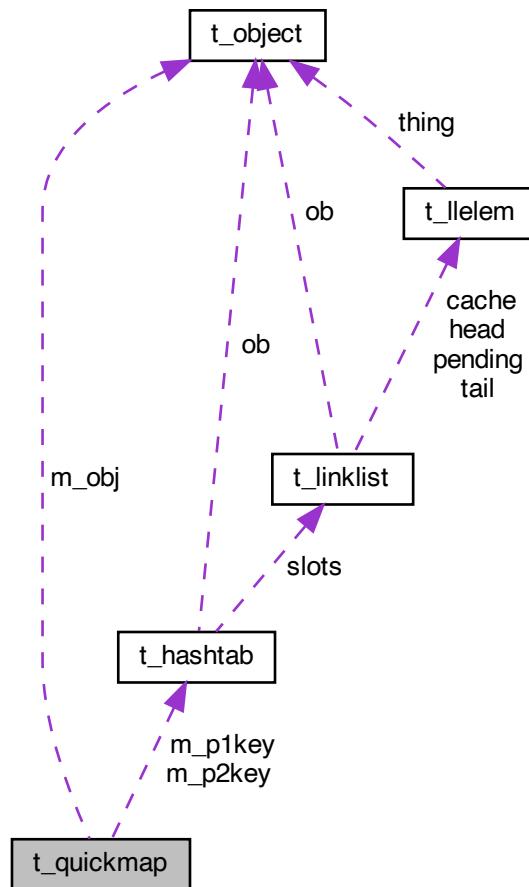
For ui objects use [t\\_pxjbox](#).

## 39.59 t\_quickmap Struct Reference

The quickmap object.

```
#include <ext_quickmap.h>
```

Collaboration diagram for t\_quickmap:



### 39.59.1 Detailed Description

The quickmap object.

This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

## 39.60 t\_rect Struct Reference

Coordinates for specifying a rectangular region.

```
#include <jpatcher_api.h>
```

### Data Fields

- double [x](#)  
*The horizontal origin.*
- double [y](#)  
*The vertical origin.*
- double [width](#)  
*The width.*
- double [height](#)  
*The height.*

### 39.60.1 Detailed Description

Coordinates for specifying a rectangular region.

See also

[t\\_pt](#)[t\\_size](#)

## 39.61 t\_signal Struct Reference

The signal data structure.

```
#include <z_dsp.h>
```

### Data Fields

- long [s\\_n](#)  
*The vector size of the signal.*
- [t\\_sample \\* s\\_vec](#)  
*A buffer holding the vector of audio samples.*
- float [s\\_sr](#)  
*The sample rate of the signal.*

### 39.61.1 Detailed Description

The signal data structure.

## 39.62 t\_size Struct Reference

Coordinates for specifying the size of a region.

```
#include <jpatcher_api.h>
```

### Data Fields

- double **width**  
*The width.*
- double **height**  
*The height.*

### 39.62.1 Detailed Description

Coordinates for specifying the size of a region.

See also

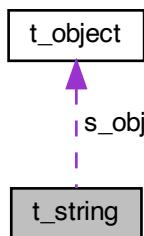
[t\\_rect](#)  
[t\\_pt](#)

## 39.63 t\_string Struct Reference

The string object.

```
#include <ext_objstring.h>
```

Collaboration diagram for t\_string:



### 39.63.1 Detailed Description

The string object.

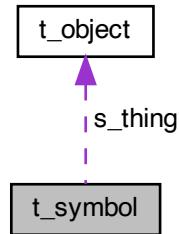
This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

## 39.64 t\_symbol Struct Reference

The symbol.

```
#include <ext_mess.h>
```

Collaboration diagram for t\_symbol:



### Data Fields

- `char * s_name`  
*name: a c-string*
- `struct object * s_thing`  
*possible binding to a `t_object`*

### 39.64.1 Detailed Description

The symbol.

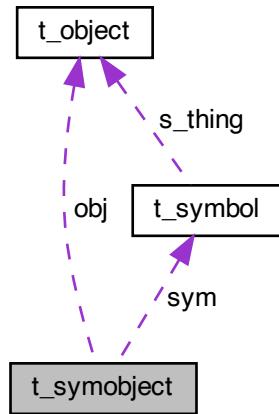
Note: You should *never* manipulate the `s_name` field of the `t_symbol` directly! Doing so will corrupt Max's symbol table. Instead, *always* use `gensym()` to get a symbol with the desired string contents for the `s_name` field.

## 39.65 t\_symobject Struct Reference

The symobject data structure.

```
#include <ext_symobject.h>
```

Collaboration diagram for t\_symobject:



### Data Fields

- `t_object obj`  
*Max object header.*
- `t_symbol * sym`  
*The symbol contained by the object.*
- `long flags`  
*Any user-flags you wish to set or get.*
- `void * thing`  
*A generic pointer for attaching additional data to the symobject.*

#### 39.65.1 Detailed Description

The symobject data structure.

## 39.66 t\_tinyobject Struct Reference

The tiny object structure sits at the head of any object to which you may pass messages (and which you may feed to `freeobject()`).

```
#include <ext_mess.h>
```

## Data Fields

- struct messlist \* **t\_messlist**  
*list of messages and methods*
- long **t\_magic**  
*magic number*

### 39.66.1 Detailed Description

The tiny object structure sits at the head of any object to which you may pass messages (and which you may feed to [freeobject\(\)](#)).

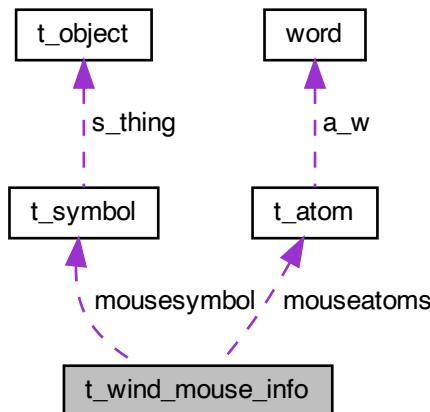
In general, you should use [t\\_object](#) instead.

## 39.67 t\_wind\_mouse\_info Struct Reference

`t_wind_mouse_info_struct` provided by `jit.window` and `jit.pwindow` mouse events

```
#include <jit.gl.common.h>
```

Collaboration diagram for `t_wind_mouse_info`:



## Data Fields

- `t_atom mouseatoms [8]`  
*h, v, (up/down), cmdKey, shiftKey, alphaLock, option, control.*
- int `argc`  
*argument count*
- `t_symbol * mousesymbol`  
*mouse event type*

### 39.67.1 Detailed Description

t\_wind\_mouse\_info\_struct provided by jit.window and jit.pwindow mouse events

## 39.68 t\_zll Struct Reference

A simple doubly-linked list used by the [t\\_funbuff](#) object.

```
#include <ext_maxtypes.h>
```

### 39.68.1 Detailed Description

A simple doubly-linked list used by the [t\\_funbuff](#) object.

## 39.69 word Union Reference

Union for packing any of the datum defined in [e\\_max\\_atomtypes](#).

```
#include <ext_mess.h>
```

### Data Fields

- [t\\_atom\\_long w\\_long](#)  
*long integer*
- [t\\_atom\\_float w\\_float](#)  
*32-bit float*
- struct symbol \* [w\\_sym](#)  
*pointer to a symbol in the Max symbol table*
- struct object \* [w\\_obj](#)  
*pointer to a [t\\_object](#) or other generic pointer*

### 39.69.1 Detailed Description

Union for packing any of the datum defined in [e\\_max\\_atomtypes](#).



# Index

A\_CANT  
    Atoms, [411](#)  
A\_COMMA  
    Atoms, [411](#)  
A\_DEFER  
    Atoms, [411](#)  
A\_DEFER\_LOW  
    Atoms, [411](#)  
A\_DEFFLOAT  
    Atoms, [411](#)  
A\_DEFLONG  
    Atoms, [410](#)  
A\_DEFSYM  
    Atoms, [411](#)  
A\_DOLLAR  
    Atoms, [411](#)  
A\_DOLLSYM  
    Atoms, [411](#)  
A\_FLOAT  
    Atoms, [410](#)  
A\_GIMME  
    Atoms, [411](#)  
A\_GIMMEBACK  
    Atoms, [411](#)  
A\_LONG  
    Atoms, [410](#)  
A\_NOTHING  
    Atoms, [410](#)  
A\_OBJ  
    Atoms, [410](#)  
A\_SEMI  
    Atoms, [411](#)  
A\_SYM  
    Atoms, [410](#)  
A\_USURP  
    Atoms, [411](#)  
A\_USURP\_LOW  
    Atoms, [411](#)  
aaCancel  
    Miscellaneous, [507](#)  
aaNo  
    Miscellaneous, [507](#)  
aaYes  
    Miscellaneous, [507](#)  
addbang

Old-Style Classes, [260](#)  
addfloat  
    Old-Style Classes, [260](#)  
addftx  
    Old-Style Classes, [260](#)  
addint  
    Old-Style Classes, [261](#)  
addinx  
    Old-Style Classes, [261](#)  
address  
    Old-Style Classes, [262](#)  
alias  
    Old-Style Classes, [262](#)  
Atom Array, [280](#)  
    atomarray\_appendatom, [282](#)  
    atomarray\_appendatoms, [283](#)  
    atomarray\_chuckindex, [283](#)  
    atomarray\_clear, [284](#)  
    atomarray\_clone, [284](#)  
    atomarray\_copyatoms, [284](#)  
    atomarray\_duplicate, [285](#)  
    atomarray\_flags, [286](#)  
    atomarray\_funall, [286](#)  
    atomarray\_getatoms, [287](#)  
    atomarray\_getflags, [287](#)  
    atomarray\_getindex, [288](#)  
    atomarray\_getsize, [288](#)  
    atomarray\_new, [289](#)  
    atomarray\_setatoms, [289](#)  
    t\_atomarray\_flags, [282](#)  
Atom Module, [851](#)  
    jit\_atom\_arg\_getdouble, [852](#)  
    jit\_atom\_arg\_getfloat, [853](#)  
    jit\_atom\_arg\_getlong, [854](#)  
    jit\_atom\_arg\_getsym, [855](#)  
    jit\_atom\_getcharfix, [856](#)  
    jit\_atom\_getfloat, [856](#)  
    jit\_atom\_getlong, [857](#)  
    jit\_atom\_getobj, [858](#)  
    jit\_atom\_getsym, [859](#)  
    jit\_atom\_setfloat, [859](#)  
    jit\_atom\_setlong, [860](#)  
    jit\_atom\_setobj, [861](#)  
    jit\_atom\_setsym, [862](#)  
atom\_alloc

Atoms, 411  
 atom\_alloc\_array  
     Atoms, 412  
 atom\_arg\_getdouble  
     Atoms, 412  
 atom\_arg\_getfloat  
     Atoms, 413  
 atom\_arg\_getlong  
     Atoms, 413  
 atom\_arg\_getobjclass  
     Atoms, 414  
 atom\_arg\_getsym  
     Atoms, 415  
 atom\_copy  
     Atoms, 415  
 atom\_getatom\_array  
     Atoms, 416  
 atom\_getchar\_array  
     Atoms, 416  
 atom\_getcharfix  
     Atoms, 417  
 atom\_getdouble\_array  
     Atoms, 417  
 atom\_getfloat  
     Atoms, 418  
 atom\_getfloat\_array  
     Atoms, 418  
 atom\_getformat  
     Atoms, 419  
 atom\_getlong  
     Atoms, 420  
 atom\_getlong\_array  
     Atoms, 420  
 atom\_getobj  
     Atoms, 421  
 atom\_getobj\_array  
     Atoms, 421  
 atom\_getobjclass  
     Atoms, 421  
 atom\_getsym  
     Atoms, 422  
 atom\_getsym\_array  
     Atoms, 422  
 atom\_gettext  
     Atoms, 423  
 atom\_gettext\_precision  
     Atoms, 424  
 atom\_gettime  
     Atoms, 424  
 atom\_setatom\_array  
     Atoms, 425  
 atom\_setchar\_array  
     Atoms, 425  
 atom\_setdouble\_array

         Atoms, 426  
 atom\_setfloat  
     Atoms, 426  
 atom\_setfloat\_array  
     Atoms, 427  
 atom\_setformat  
     Atoms, 427  
 atom\_setlong  
     Atoms, 428  
 atom\_setlong\_array  
     Atoms, 428  
 atom\_setobj  
     Atoms, 429  
 atom\_setobj\_array  
     Atoms, 429  
 atom\_setparse  
     Atoms, 430  
 atom\_setsym  
     Atoms, 431  
 atom\_setsym\_array  
     Atoms, 431  
 atomarray\_appendatom  
     Atom Array, 282  
 atomarray\_appendatoms  
     Atom Array, 283  
 atomarray\_chuckindex  
     Atom Array, 283  
 atomarray\_clear  
     Atom Array, 284  
 atomarray\_clone  
     Atom Array, 284  
 atomarray\_copyatoms  
     Atom Array, 284  
 atomarray\_duplicate  
     Atom Array, 285  
 atomarray\_flags  
     Atom Array, 286  
 atomarray\_funall  
     Atom Array, 286  
 atomarray\_getatoms  
     Atom Array, 287  
 atomarray\_getflags  
     Atom Array, 287  
 atomarray\_getindex  
     Atom Array, 288  
 atomarray\_getsize  
     Atom Array, 288  
 atomarray\_new  
     Atom Array, 289  
 atomarray\_setatoms  
     Atom Array, 289  
 atombuf\_free  
     Atombufs, 434  
 atombuf\_new

Atombufs, 434  
atombuf\_text  
    Atombufs, 435  
Atombufs, 433  
    atombuf\_free, 434  
    atombuf\_new, 434  
    atombuf\_text, 435  
atomisatomarray  
    Atoms, 432  
atomisdictionary  
    Atoms, 432  
atomisstring  
    Atoms, 432  
Atoms, 407  
    A\_CANT, 411  
    A\_COMMA, 411  
    A\_DEFER, 411  
    A\_DEFER\_LOW, 411  
    A\_DEFFLOAT, 411  
    A\_DEFLONG, 410  
    A\_DEFSYM, 411  
    A\_DOLLAR, 411  
    A\_DOLLSYM, 411  
    A\_FLOAT, 410  
    A\_GIMME, 411  
    A\_GIMMEBACK, 411  
    A\_LONG, 410  
    A\_NOTHING, 410  
    A\_OBJ, 410  
    A\_SEMI, 411  
    A\_SYM, 410  
    A\_USURP, 411  
    A\_USURP\_LOW, 411  
atom\_alloc, 411  
atom\_alloc\_array, 412  
atom\_arg\_getdouble, 412  
atom\_arg\_getfloat, 413  
atom\_arg\_getlong, 413  
atom\_arg\_getobjclass, 414  
atom\_arg\_getsym, 415  
atom\_copy, 415  
atom\_getatom\_array, 416  
atom\_getchar\_array, 416  
atom\_getcharfix, 417  
atom\_getdouble\_array, 417  
atom\_getfloat, 418  
atom\_getfloat\_array, 418  
atom\_getformat, 419  
atom\_getlong, 420  
atom\_getlong\_array, 420  
atom\_getobj, 421  
atom\_getobj\_array, 421  
atom\_getobjclass, 421  
atom\_getsym, 422  
atom\_getsym\_array, 422  
atom\_gettext, 423  
atom\_gettext\_precision, 424  
atom\_gettime, 424  
atom\_setatom\_array, 425  
atom\_setchar\_array, 425  
atom\_setdouble\_array, 426  
atom\_setfloat, 426  
atom\_setfloat\_array, 427  
atom\_setformat, 427  
atom\_setlong, 428  
atom\_setlong\_array, 428  
atom\_setobj, 429  
atom\_setobj\_array, 429  
atom\_setparse, 430  
atom\_setsym, 431  
atom\_setsym\_array, 431  
atomisatomarray, 432  
atomisdictionary, 432  
atomisstring, 432  
e\_max\_atom\_gettext\_flags, 409  
e\_max\_atomtypes, 410  
OBEX\_UTIL\_ATOM\_GETTEXT\_COMMAS\_DELIM,  
    410  
OBEX\_UTIL\_ATOM\_GETTEXT\_DEFAULT, 410  
OBEX\_UTIL\_ATOM\_GETTEXT\_FORCE\_ZEROS,  
    410  
OBEX\_UTIL\_ATOM\_GETTEXT\_LINEBREAK\_NODELIM,  
    410  
OBEX\_UTIL\_ATOM\_GETTEXT\_NOESCAPE, 410  
OBEX\_UTIL\_ATOM\_GETTEXT\_NUM\_HI\_RES, 410  
OBEX\_UTIL\_ATOM\_GETTEXT\_NUM\_LO\_RES,  
    410  
OBEX\_UTIL\_ATOM\_GETTEXT\_SYM\_FORCE\_QUOTE,  
    410  
OBEX\_UTIL\_ATOM\_GETTEXT\_SYM\_NO\_QUOTE,  
    410  
OBEX\_UTIL\_ATOM\_GETTEXT\_TRUNCATE\_ZEROS,  
    410  
postargs, 433  
atoms\_to\_jrgba  
    Colors, 802  
attr\_addfilter\_clip  
    Attributes, 215  
attr\_addfilter\_clip\_scale  
    Attributes, 216  
attr\_addfilterget\_clip  
    Attributes, 216  
attr\_addfilterget\_clip\_scale  
    Attributes, 217  
attr\_addfilterget\_proc  
    Attributes, 218  
attr\_addfilterset\_clip  
    Attributes, 218

attr\_addfilterset\_clip\_scale  
     Attributes, 219  
 attr\_addfilterset\_proc  
     Attributes, 219  
 attr\_args\_dictionary  
     Attributes, 220  
 attr\_args\_offset  
     Attributes, 221  
 attr\_args\_process  
     Attributes, 221  
 attr\_dictionary\_check  
     Attributes, 222  
 attr\_dictionary\_process  
     Attributes, 222  
 ATTR\_FLAGS\_NONE  
     Attributes, 215  
 ATTR\_GET\_OPAQUE  
     Attributes, 215  
 ATTR\_GET\_OPAQUE\_USER  
     Attributes, 215  
 attr\_offset\_array\_new  
     Attributes, 223  
 attr\_offset\_new  
     Attributes, 224  
 ATTR\_SET\_OPAQUE  
     Attributes, 215  
 ATTR\_SET\_OPAQUE\_USER  
     Attributes, 215  
 Attribute Module, 863  
     jit\_attr\_canget, 865  
     jit\_attr\_canset, 865  
     jit\_attr\_filter\_clip\_new, 866  
     jit\_attr\_filter\_proc\_new, 866  
     jit\_attr\_filterget, 867  
     jit\_attr\_filterset, 868  
     jit\_attr\_get, 868  
     jit\_attr\_getchar\_array, 869  
     jit\_attr\_getdouble\_array, 870  
     jit\_attr\_getfloat, 871  
     jit\_attr\_getfloat\_array, 872  
     jit\_attr\_getlong, 873  
     jit\_attr\_getlong\_array, 874  
     jit\_attr\_getmethod, 875  
     jit\_attr\_getname, 875  
     jit\_attr\_getsym, 876  
     jit\_attr\_getsym\_array, 877  
     jit\_attr\_gettype, 878  
     jit\_attr\_offset\_array\_new, 878  
     jit\_attr\_offset\_new, 879  
     jit\_attr\_set, 880  
     jit\_attr\_setchar\_array, 881  
     jit\_attr\_setdouble\_array, 882  
     jit\_attr\_setfloat, 883  
     jit\_attr\_setfloat\_array, 884  
     jit\_attr\_setlong, 885  
     jit\_attr\_setlong\_array, 886  
     jit\_attr\_setsym, 887  
     jit\_attr\_setsym\_array, 888  
     jit\_attr\_symcompare, 889  
     jit\_attr\_usercanget, 890  
     jit\_attr\_usercanset, 890  
     jit\_attribute\_new, 891  
 attribute\_new  
     Attributes, 225  
 Attributes, 151  
     attr\_addfilter\_clip, 215  
     attr\_addfilter\_clip\_scale, 216  
     attr\_addfilterget\_clip, 216  
     attr\_addfilterget\_clip\_scale, 217  
     attr\_addfilterget\_proc, 218  
     attr\_addfilterset\_clip, 218  
     attr\_addfilterset\_clip\_scale, 219  
     attr\_addfilterset\_proc, 219  
     attr\_args\_dictionary, 220  
     attr\_args\_offset, 221  
     attr\_args\_process, 221  
     attr\_dictionary\_check, 222  
     attr\_dictionary\_process, 222  
     ATTR\_FLAGS\_NONE, 215  
     ATTR\_GET\_OPAQUE, 215  
     ATTR\_GET\_OPAQUE\_USER, 215  
     attr\_offset\_array\_new, 223  
     attr\_offset\_new, 224  
     ATTR\_SET\_OPAQUE, 215  
     ATTR\_SET\_OPAQUE\_USER, 215  
     attribute\_new, 225  
     CLASS\_ATTR\_ACCESSORS, 160  
     CLASS\_ATTR\_ADD\_FLAGS, 160  
     CLASS\_ATTR\_ALIAS, 161  
     CLASS\_ATTR\_ATOM, 161  
     CLASS\_ATTR\_ATOM\_ARRAY, 161  
     CLASS\_ATTR\_ATOM\_LONG, 162  
     CLASS\_ATTR\_ATOM\_LONG\_ARRAY, 162  
     CLASS\_ATTR\_ATOM\_VARSIZE, 163  
     CLASS\_ATTR\_BASIC, 163  
     CLASS\_ATTR\_CATEGORY, 164  
     CLASS\_ATTR\_CHAR, 164  
     CLASS\_ATTR\_CHAR\_ARRAY, 165  
     CLASS\_ATTR\_CHAR\_VARSIZE, 165  
     CLASS\_ATTR\_DEFAULT, 166  
     CLASS\_ATTR\_DEFAULT\_PAINT, 166  
     CLASS\_ATTR\_DEFAULT\_SAVE, 167  
     CLASS\_ATTR\_DEFAULT\_SAVE\_PAINT, 167  
     CLASS\_ATTR\_DEFAULTNAME, 168  
     CLASS\_ATTR\_DEFAULTNAME\_PAINT, 168  
     CLASS\_ATTR\_DEFAULTNAME\_SAVE, 170  
     CLASS\_ATTR\_DEFAULTNAME\_SAVE\_PAINT, 170  
     CLASS\_ATTR\_DOUBLE, 172

CLASS\_ATTR\_DOUBLE\_ARRAY, 172  
CLASS\_ATTR\_DOUBLE\_VARSIZE, 174  
CLASS\_ATTR\_ENUM, 174  
CLASS\_ATTR\_ENUMINDEX, 175  
CLASS\_ATTR\_FILTER\_CLIP, 175  
CLASS\_ATTR\_FILTER\_MAX, 176  
CLASS\_ATTR\_FILTER\_MIN, 177  
CLASS\_ATTR\_FLOAT, 177  
CLASS\_ATTR\_FLOAT\_ARRAY, 178  
CLASS\_ATTR\_FLOAT\_VARSIZE, 178  
CLASS\_ATTR\_INT32, 179  
CLASS\_ATTR\_INTRODUCED, 179  
CLASS\_ATTR\_INVISIBLE, 179  
CLASS\_ATTR\_LABEL, 180  
CLASS\_ATTR\_LEGACYDEFAULT, 180  
CLASS\_ATTR\_LONG, 181  
CLASS\_ATTR\_LONG\_ARRAY, 181  
CLASS\_ATTR\_LONG\_VARSIZE, 182  
CLASS\_ATTR\_MAX, 182  
CLASS\_ATTR\_MIN, 183  
CLASS\_ATTR\_OBJ, 183  
CLASS\_ATTR\_OBJ\_ARRAY, 184  
CLASS\_ATTR\_OBJ\_VARSIZE, 184  
CLASS\_ATTR\_OBSOLETE, 185  
CLASS\_ATTR\_OFFSET\_DUMMY, 185  
CLASS\_ATTR\_ORDER, 186  
CLASS\_ATTR\_PAINT, 186  
CLASS\_ATTR\_REMOVE\_FLAGS, 187  
CLASS\_ATTR\_RENAMED, 187  
CLASS\_ATTR\_RGBA, 188  
CLASS\_ATTR\_SAVE, 188  
CLASS\_ATTR\_SELFSAVE, 189  
CLASS\_ATTR\_STYLE, 189  
CLASS\_ATTR\_STYLE\_LABEL, 190  
CLASS\_ATTR\_SYM, 190  
CLASS\_ATTR\_SYM\_ARRAY, 191  
CLASS\_ATTR\_SYM\_VARSIZE, 191  
CLASS\_METHOD\_ATTR\_PARSE, 192  
CLASS\_METHOD\_INTRODUCED, 192  
CLASS\_METHOD\_OBSOLETE, 193  
CLASS\_METHOD\_RENAMED, 193  
CLASS\_STICKY\_ATTR, 194  
CLASS\_STICKY\_ATTR\_CLEAR, 194  
CLASS\_STICKY\_METHOD, 195  
CLASS\_STICKY\_METHOD\_CLEAR, 195  
e\_max\_attrflags, 215  
OBJ\_ATTR\_ATOM, 196  
OBJ\_ATTR\_ATOM\_ARRAY, 196  
OBJ\_ATTR\_CHAR, 197  
OBJ\_ATTR\_CHAR\_ARRAY, 197  
OBJ\_ATTR\_DEFAULT, 198  
OBJ\_ATTR\_DEFAULT\_SAVE, 198  
OBJ\_ATTR\_DOUBLE, 198  
OBJ\_ATTR\_DOUBLE\_ARRAY, 199  
OBJ\_ATTR\_FLOAT, 199  
OBJ\_ATTR\_FLOAT\_ARRAY, 200  
OBJ\_ATTR\_LONG, 200  
OBJ\_ATTR\_LONG\_ARRAY, 201  
OBJ\_ATTR\_OBJ, 201  
OBJ\_ATTR\_OBJ\_ARRAY, 202  
OBJ\_ATTR\_SAVE, 202  
OBJ\_ATTR\_SYM, 202  
OBJ\_ATTR\_SYM\_ARRAY, 203  
object\_addattr, 226  
object\_attr\_get, 226  
object\_attr\_get\_rect, 227  
object\_attr\_getchar\_array, 227  
object\_attr\_getcolor, 228  
object\_attr\_getdouble\_array, 229  
object\_attr\_getdump, 229  
object\_attr\_getfloat, 230  
object\_attr\_getfloat\_array, 230  
object\_attr\_getjrgba, 231  
object\_attr\_getlong, 231  
object\_attr\_getlong\_array, 232  
object\_attr\_getpt, 233  
object\_attr\_getsize, 233  
object\_attr\_getsym, 234  
object\_attr\_getsym\_array, 234  
object\_attr\_method, 235  
object\_attr\_set\_rect, 235  
object\_attr\_set\_xywh, 236  
object\_attr\_setchar\_array, 236  
object\_attr\_setcolor, 237  
object\_attr\_setdouble\_array, 237  
object\_attr\_setfloat, 238  
object\_attr\_setfloat\_array, 239  
object\_attr\_setjrgba, 239  
object\_attr\_setlong, 240  
object\_attr\_setlong\_array, 240  
object\_attr\_setparse, 241  
object\_attr\_setpt, 241  
object\_attr\_setsize, 242  
object\_attr\_setsym, 242  
object\_attr\_setsym\_array, 243  
object\_attr\_setvalueof, 243  
object\_attr\_usercanget, 244  
object\_attr\_usercanset, 244  
object\_chuckattr, 245  
object\_deleteattr, 245  
object\_new\_parse, 246  
STRUCT\_ATTR\_ATOM, 203  
STRUCT\_ATTR\_ATOM\_ARRAY, 204  
STRUCT\_ATTR\_ATOM\_LONG, 204  
STRUCT\_ATTR\_ATOM\_VARSIZE, 205  
STRUCT\_ATTR\_CHAR, 205  
STRUCT\_ATTR\_CHAR\_ARRAY, 206  
STRUCT\_ATTR\_CHAR\_VARSIZE, 206

STRUCT\_ATTR\_DOUBLE, 207  
 STRUCT\_ATTR\_DOUBLE\_ARRAY, 207  
 STRUCT\_ATTR\_DOUBLE\_VARSIZE, 208  
 STRUCT\_ATTR\_FLOAT, 208  
 STRUCT\_ATTR\_FLOAT\_ARRAY, 209  
 STRUCT\_ATTR\_FLOAT\_VARSIZE, 209  
 STRUCT\_ATTR\_LONG, 210  
 STRUCT\_ATTR\_LONG\_ARRAY, 210  
 STRUCT\_ATTR\_LONG\_VARSIZE, 211  
 STRUCT\_ATTR\_OBJ, 212  
 STRUCT\_ATTR\_OBJ\_ARRAY, 212  
 STRUCT\_ATTR\_OBJ\_VARSIZE, 212  
 STRUCT\_ATTR\_SYM, 213  
 STRUCT\_ATTR\_SYM\_ARRAY, 213  
 STRUCT\_ATTR\_SYM\_VARSIZE, 214

**b\_dirty\_clock**  
 t\_buffer, 1187

**bangout**  
 Inlets and Outlets, 269

**BEGIN\_USING\_C\_LINKAGE**  
 Miscellaneous, 504

**Binary Module**, 892
 

- jit\_bin\_read\_chunk\_info, 892
- jit\_bin\_read\_header, 893
- jit\_bin\_read\_matrix, 894
- jit\_bin\_write\_header, 895
- jit\_bin\_write\_matrix, 896

**binbuf\_append**  
 Binbufs, 436

**binbuf\_eval**  
 Binbufs, 437

**binbuf\_getatom**  
 Binbufs, 437

**binbuf\_insert**  
 Binbufs, 438

**binbuf\_new**  
 Binbufs, 438

**binbuf\_set**  
 Binbufs, 439

**binbuf\_text**  
 Binbufs, 439

**binbuf\_totext**  
 Binbufs, 440

**binbuf\_vinsert**  
 Binbufs, 440

**Binbufs**, 435
 

- binbuf\_append, 436
- binbuf\_eval, 437
- binbuf\_getatom, 437
- binbuf\_insert, 438
- binbuf\_new, 438
- binbuf\_set, 439
- binbuf\_text, 439

**binbuf\_totext**, 440

**binbuf\_vinsert**, 440

**Box Layer**, 842
 

- jbox\_end\_layer, 843
- jbox\_invalidate\_layer, 843
- jbox\_paint\_layer, 844
- jbox\_start\_layer, 844

**buffer\_getchannelcount**  
 Buffers, 558

**buffer\_getfilename**  
 Buffers, 559

**buffer\_getframecount**  
 Buffers, 559

**buffer\_getmillisamplerate**  
 Buffers, 560

**buffer\_getsamplerate**  
 Buffers, 560

**buffer\_locksamples**  
 Buffers, 560

**buffer\_ref\_exists**  
 Buffers, 562

**buffer\_ref\_getobject**  
 Buffers, 562

**buffer\_ref\_new**  
 Buffers, 562

**buffer\_ref\_notify**  
 Buffers, 563

**buffer\_ref\_set**  
 Buffers, 563

**buffer\_setdirty**  
 Buffers, 564

**buffer\_setpadding**  
 Buffers, 564

**buffer\_unlocksamples**  
 Buffers, 565

**buffer\_view**  
 Buffers, 565

**Buffers**, 556
 

- buffer\_getchannelcount, 558
- buffer\_getfilename, 559
- buffer\_getframecount, 559
- buffer\_getmillisamplerate, 560
- buffer\_getsamplerate, 560
- buffer\_locksamples, 560
- buffer\_ref\_exists, 562
- buffer\_ref\_getobject, 562
- buffer\_ref\_new, 562
- buffer\_ref\_notify, 563
- buffer\_ref\_set, 563
- buffer\_setdirty, 564
- buffer\_setpadding, 564
- buffer\_unlocksamples, 565
- buffer\_view, 565

t\_buffer\_obj, 558  
t\_buffer\_ref, 558  
Byte Ordering, 522  
BYTEORDER\_SWAPF32, 523  
BYTEORDER\_SWAPF64, 524  
BYTEORDER\_SWAPW16, 524  
BYTEORDER\_SWAPW32, 524  
BYTEORDER\_SWAPW64, 525  
BYTEORDER\_SWAPF32  
    Byte Ordering, 523  
BYTEORDER\_SWAPF64  
    Byte Ordering, 524  
BYTEORDER\_SWAPW16  
    Byte Ordering, 524  
BYTEORDER\_SWAPW32  
    Byte Ordering, 524  
BYTEORDER\_SWAPW64  
    Byte Ordering, 525

calcoffset  
    Miscellaneous, 504  
charset\_convert  
    Unicode, 848  
charset unicodetouft8  
    Unicode, 849  
charset\_utf8\_count  
    Unicode, 849  
charset\_utf8\_offset  
    Unicode, 850  
charset\_utf8tounicode  
    Unicode, 850  
CLAMP  
    Miscellaneous, 504  
Class Module, 897  
    class\_copy, 898  
    jit\_class\_adddornment, 899  
    jit\_class\_addattr, 899  
    jit\_class\_addinterface, 900  
    jit\_class\_addmethod, 901  
    jit\_class\_addtypedwrapper, 902  
    jit\_class\_adornment\_get, 903  
    jit\_class\_attr\_get, 903  
    jit\_class\_findbyname, 904  
    jit\_class\_free, 904  
    jit\_class\_mess, 905  
    jit\_class\_method, 906  
    jit\_class\_method\_addargsafe, 906  
    jit\_class\_method\_argsafe\_get, 907  
    jit\_class\_nameget, 908  
    jit\_class\_new, 908  
    jit\_class\_register, 909  
    jit\_class\_symcompare, 910  
    jit\_class\_typedwrapper\_get, 911

class\_addattr  
    Classes, 250  
    class\_addmethod  
        Classes, 250  
    class\_alias  
        Classes, 251  
    CLASS\_ATTR\_ACCESSORS  
        Attributes, 160  
    CLASS\_ATTR\_ADD\_FLAGS  
        Attributes, 160  
    CLASS\_ATTR\_ALIAS  
        Attributes, 161  
    CLASS\_ATTR\_ATOM  
        Attributes, 161  
    CLASS\_ATTR\_ATOM\_ARRAY  
        Attributes, 161  
    CLASS\_ATTR\_ATOM\_LONG  
        Attributes, 162  
    CLASS\_ATTR\_ATOM\_LONG\_ARRAY  
        Attributes, 162  
    CLASS\_ATTR\_ATOM\_VARSIZE  
        Attributes, 163  
    CLASS\_ATTR\_BASIC  
        Attributes, 163  
    CLASS\_ATTR\_CATEGORY  
        Attributes, 164  
    CLASS\_ATTR\_CHAR  
        Attributes, 164  
    CLASS\_ATTR\_CHAR\_ARRAY  
        Attributes, 165  
    CLASS\_ATTR\_CHAR\_VARSIZE  
        Attributes, 165  
    CLASS\_ATTR\_DEFAULT  
        Attributes, 166  
    CLASS\_ATTR\_DEFAULT\_PAINT  
        Attributes, 166  
    CLASS\_ATTR\_DEFAULT\_SAVE  
        Attributes, 167  
    CLASS\_ATTR\_DEFAULT\_SAVE\_PAINT  
        Attributes, 167  
    CLASS\_ATTR\_DEFAULTNAME  
        Attributes, 168  
    CLASS\_ATTR\_DEFAULTNAME\_PAINT  
        Attributes, 168  
    CLASS\_ATTR\_DEFAULTNAME\_SAVE  
        Attributes, 170  
    CLASS\_ATTR\_DEFAULTNAME\_SAVE\_PAINT  
        Attributes, 170  
    CLASS\_ATTR\_DOUBLE  
        Attributes, 172  
    CLASS\_ATTR\_DOUBLE\_ARRAY  
        Attributes, 172  
    CLASS\_ATTR\_DOUBLE\_VARSIZE  
        Attributes, 174  
    CLASS\_ATTR\_ENUM

Attributes, 174  
**CLASS\_ATTR\_ENUMINDEX**  
 Attributes, 175  
**CLASS\_ATTR\_FILTER\_CLIP**  
 Attributes, 175  
**CLASS\_ATTR\_FILTER\_MAX**  
 Attributes, 176  
**CLASS\_ATTR\_FILTER\_MIN**  
 Attributes, 177  
**CLASS\_ATTR\_FLOAT**  
 Attributes, 177  
**CLASS\_ATTR\_FLOAT\_ARRAY**  
 Attributes, 178  
**CLASS\_ATTR\_FLOAT\_VARSIZE**  
 Attributes, 178  
**CLASS\_ATTR\_INT32**  
 Attributes, 179  
**CLASS\_ATTR\_INTRODUCED**  
 Attributes, 179  
**CLASS\_ATTR\_INVISIBLE**  
 Attributes, 179  
**CLASS\_ATTR\_LABEL**  
 Attributes, 180  
**CLASS\_ATTR\_LEGACYDEFAULT**  
 Attributes, 180  
**CLASS\_ATTR\_LONG**  
 Attributes, 181  
**CLASS\_ATTR\_LONG\_ARRAY**  
 Attributes, 181  
**CLASS\_ATTR\_LONG\_VARSIZE**  
 Attributes, 182  
**CLASS\_ATTR\_MAX**  
 Attributes, 182  
**CLASS\_ATTR\_MIN**  
 Attributes, 183  
**CLASS\_ATTR\_OBJ**  
 Attributes, 183  
**CLASS\_ATTR\_OBJ\_ARRAY**  
 Attributes, 184  
**CLASS\_ATTR\_OBJ\_VARSIZE**  
 Attributes, 184  
**CLASS\_ATTR\_OBSOLETE**  
 Attributes, 185  
**CLASS\_ATTR\_OFFSET\_DUMMY**  
 Attributes, 185  
**CLASS\_ATTR\_ORDER**  
 Attributes, 186  
**CLASS\_ATTR\_PAINT**  
 Attributes, 186  
**CLASS\_ATTR\_REMOVE\_FLAGS**  
 Attributes, 187  
**CLASS\_ATTR\_RENAMED**  
 Attributes, 187  
**CLASS\_ATTR\_RGBA**  
 Attributes, 188  
**CLASS\_ATTR\_SAVE**  
 Attributes, 188  
**CLASS\_ATTR\_SELFSAVE**  
 Attributes, 189  
**class\_attr\_setfill**  
 Styles, 810  
**class\_attr\_setstyle**  
 Styles, 810  
**CLASS\_ATTR\_STYLE**  
 Attributes, 189  
**class\_attr\_style\_alias**  
 Styles, 811  
**CLASS\_ATTR\_STYLE\_ALIAS\_COMPATIBILITY**  
 Styles, 806  
**CLASS\_ATTR\_STYLE\_ALIAS\_NOSAVE**  
 Styles, 807  
**CLASS\_ATTR\_STYLE\_ALIAS\_RGBA\_LEGACY**  
 Styles, 807  
**CLASS\_ATTR\_STYLE\_LABEL**  
 Attributes, 190  
**CLASS\_ATTR\_STYLE\_RGBA**  
 Styles, 808  
**CLASS\_ATTR\_STYLE\_RGBA\_NOSAVE**  
 Styles, 808  
**CLASS\_ATTR\_STYLE\_RGBA\_PREVIEW**  
 Styles, 809  
**class\_attr\_stlemap**  
 Styles, 811  
**CLASS\_ATTR\_SYM**  
 Attributes, 190  
**CLASS\_ATTR\_SYM\_ARRAY**  
 Attributes, 191  
**CLASS\_ATTR\_SYM\_VARSIZE**  
 Attributes, 191  
**CLASS\_BOX**  
 Classes, 249  
**class\_copy**  
 Class Module, 898  
**class\_dspinit**  
 MSP, 551  
**class\_dspinitbbox**  
 MSP, 552  
**class\_dumpout\_wrap**  
 Classes, 251  
**class\_findbyname**  
 Classes, 252  
**class\_findbyname\_casfree**  
 Classes, 252  
**CLASS\_FLAG\_ALIAS**  
 Classes, 249  
**CLASS\_FLAG\_BOX**  
 Classes, 249  
**CLASS\_FLAG\_DO\_NOT\_PARSE\_ATTR\_ARGS**

Classes, 249  
CLASS\_FLAG\_DO\_NOT\_ZERO  
    Classes, 249  
CLASS\_FLAG\_DYNAMICCOLOR  
    Classes, 249  
CLASS\_FLAG\_MULTITOUCH  
    Classes, 249  
CLASS\_FLAG\_NEWDICTIONARY  
    Classes, 249  
CLASS\_FLAG\_NOATTRIBUTES  
    Classes, 249  
CLASS\_FLAG\_OBJECT\_METHOD  
    Classes, 249  
CLASS\_FLAG\_OWN\_DATA  
    Classes, 249  
CLASS\_FLAG\_OWNATTRIBUTES  
    Classes, 249  
CLASS\_FLAG\_PARAMETER  
    Classes, 249  
CLASS\_FLAG\_POLYGLOT  
    Classes, 249  
CLASS\_FLAG\_REGISTERED  
    Classes, 249  
CLASS\_FLAG RETYPEABLE  
    Classes, 249  
CLASS\_FLAG\_UIOBJECT  
    Classes, 249  
CLASS\_FLAGUSES\_PROXYIES  
    Classes, 249  
CLASS\_FLAG\_VISUALIZER  
    Classes, 249  
class\_free  
    Classes, 253  
class\_is\_ui  
    Classes, 253  
CLASS\_METHOD\_ATTR\_PARSE  
    Attributes, 192  
CLASS\_METHOD\_INTRODUCED  
    Attributes, 192  
CLASS\_METHOD\_OBSOLETE  
    Attributes, 193  
CLASS\_METHOD\_RENAMED  
    Attributes, 193  
class\_nameget  
    Classes, 253  
class\_new  
    Classes, 254  
class\_obexoffset\_get  
    Classes, 255  
class\_obexoffset\_set  
    Classes, 255  
class\_register  
    Classes, 255  
class\_setname

Old-Style Classes, 262  
CLASS\_STICKY\_ATTR  
    Attributes, 194  
CLASS\_STICKY\_ATTR\_CLEAR  
    Attributes, 194  
CLASS\_STICKY\_METHOD  
    Attributes, 195  
CLASS\_STICKY\_METHOD\_CLEAR  
    Attributes, 195  
class\_subclass  
    Classes, 256  
class\_super\_construct  
    Classes, 256  
class\_time\_addattr  
    ITM Time Objects, 700  
Classes, 247  
    class\_addattr, 250  
    class\_addmethod, 250  
    class\_alias, 251  
    CLASS\_BOX, 249  
    class\_dumpout\_wrap, 251  
    class\_findbyname, 252  
    class\_findbyname\_casemode, 252  
    CLASS\_FLAG\_ALIAS, 249  
    CLASS\_FLAG\_BOX, 249  
    CLASS\_FLAG\_DO\_NOT\_PARSE\_ATTR\_ARGS,  
        249  
    CLASS\_FLAG\_DO\_NOT\_ZERO, 249  
    CLASS\_FLAG\_DYNAMICCOLOR, 249  
    CLASS\_FLAG\_MULTITOUCH, 249  
    CLASS\_FLAG\_NEWDICTIONARY, 249  
    CLASS\_FLAG\_NOATTRIBUTES, 249  
    CLASS\_FLAG\_OBJECT\_METHOD, 249  
    CLASS\_FLAG\_OWN\_DATA, 249  
    CLASS\_FLAG\_OWNATTRIBUTES, 249  
    CLASS\_FLAG\_PARAMETER, 249  
    CLASS\_FLAG\_POLYGLOT, 249  
    CLASS\_FLAG\_REGISTERED, 249  
    CLASS\_FLAG RETYPEABLE, 249  
    CLASS\_FLAG\_UIOBJECT, 249  
    CLASS\_FLAGUSES\_PROXYIES, 249  
    CLASS\_FLAG\_VISUALIZER, 249  
    class\_free, 253  
    class\_is\_ui, 253  
    class\_nameget, 253  
    class\_new, 254  
    class\_obexoffset\_get, 255  
    class\_obexoffset\_set, 255  
    class\_register, 255  
    class\_subclass, 256  
    class\_super\_construct, 256  
    e\_max\_class\_flags, 249  
    ext\_main, 257  
    classname\_openhelp

Objects, 571  
 classname\_openquery  
     Objects, 571  
 classname\_openrefpage  
     Objects, 572  
 clock\_delay  
     Clocks, 679  
 clock\_fset  
     Clocks, 680  
 clock\_gettime  
     Clocks, 680  
 clock\_new  
     Clocks, 680  
 clock\_unset  
     Clocks, 681  
 Clocks, 675  
     clock\_delay, 679  
     clock\_fset, 680  
     clock\_gettime, 680  
     clock\_new, 680  
     clock\_unset, 681  
     gettime, 681  
     gettime\_forobject, 682  
     scheduler\_fromobject, 682  
     scheduler\_get, 682  
     scheduler\_gettime, 683  
     scheduler\_new, 683  
     scheduler\_run, 684  
     scheduler\_set, 684  
     scheduler\_settime, 684  
     scheduler\_shift, 685  
     setclock\_delay, 685  
     setclock\_fdelay, 686  
     setclock\_gettime, 686  
     setclock\_gettime, 687  
     setclock\_unset, 687  
     systimer\_gettime, 688  
 Colors, 801  
     atoms\_to\_jrgba, 802  
     jrgba\_attr\_get, 803  
     jrgba\_attr\_set, 803  
     jrgba\_compare, 804  
     jrgba\_copy, 804  
     jrgba\_set, 804  
     jrgba\_to\_atoms, 805  
 Console, 516  
     cpost, 517  
     error, 517  
     object\_error, 518  
     object\_error\_obtrusive, 519  
     object\_post, 519  
     object\_warn, 520  
     ouchstring, 520  
     post, 521  
     postatom, 522  
 cpost  
     Console, 517  
 Critical Regions, 728  
     critical\_enter, 730  
     critical\_exit, 730  
     critical\_free, 731  
     critical\_new, 731  
     critical\_tryenter, 731  
 critical\_enter  
     Critical Regions, 730  
 critical\_exit  
     Critical Regions, 730  
 critical\_free  
     Critical Regions, 731  
 critical\_new  
     Critical Regions, 731  
 critical\_tryenter  
     Critical Regions, 731  
 Data Storage, 278  
     e\_max\_datastore\_flags, 280  
     OBJ\_FLAG\_CLONE, 280  
     OBJ\_FLAG\_DANGER, 280  
     OBJ\_FLAG\_DATA, 280  
     OBJ\_FLAG\_DEBUG, 280  
     OBJ\_FLAG\_INHERITABLE, 280  
     OBJ\_FLAG\_ITERATING, 280  
     OBJ\_FLAG\_MEMORY, 280  
     OBJ\_FLAG\_OBJ, 280  
     OBJ\_FLAG\_REF, 280  
     OBJ\_FLAG\_SILENT, 280  
     t\_cmpfn, 279  
 Data Types, 405  
 Database, 290  
     db\_close, 293  
     db\_open, 293  
     db\_open\_ext, 294  
     db\_query, 294  
     db\_query\_direct, 295  
     db\_query\_getlastinsertid, 295  
     db\_query\_silent, 295  
     db\_query\_table\_addcolumn, 296  
     db\_query\_table\_new, 297  
     db\_result\_clear, 297  
     db\_result\_datetimeinseconds, 297  
     db\_result\_fieldname, 298  
     db\_result\_float, 298  
     db\_result\_long, 299  
     db\_result\_nextrecord, 299  
     db\_result\_numfields, 300  
     db\_result\_numrecords, 300  
     db\_result\_reset, 300  
     db\_result\_string, 301

db\_transaction\_end, 301  
db\_transaction\_flush, 302  
db\_transaction\_start, 302  
db\_util\_datetostring, 302  
db\_util\_stringtodate, 303  
db\_view\_create, 303  
db\_view\_getresult, 304  
db\_view\_remove, 304  
db\_view\_setquery, 304  
db\_vquery, 305  
t\_database, 292  
t\_db\_result, 292  
t\_db\_view, 292  
DataView, 845  
    jdataview\_getclient, 846  
    jdataview\_new, 846  
    jdataview\_setclient, 847  
db\_close  
    Database, 293  
db\_open  
    Database, 293  
db\_open\_ext  
    Database, 294  
db\_query  
    Database, 294  
db\_query\_direct  
    Database, 295  
db\_query\_getlastinsertid  
    Database, 295  
db\_query\_silent  
    Database, 295  
db\_query\_table\_addcolumn  
    Database, 296  
db\_query\_table\_new  
    Database, 297  
db\_result\_clear  
    Database, 297  
db\_result\_datetimeinseconds  
    Database, 297  
db\_result\_fieldname  
    Database, 298  
db\_result\_float  
    Database, 298  
db\_result\_long  
    Database, 299  
db\_result\_nextrecord  
    Database, 299  
db\_result\_numfields  
    Database, 300  
db\_result\_numrecords  
    Database, 300  
db\_result\_reset  
    Database, 300  
db\_result\_string  
    Database, 301  
db\_transaction\_end  
    Database, 301  
db\_transaction\_flush  
    Database, 302  
db\_transaction\_start  
    Database, 302  
db\_util\_datetostring  
    Database, 302  
db\_util\_stringtodate  
    Database, 303  
db\_view\_create  
    Database, 303  
db\_view\_getresult  
    Database, 304  
db\_view\_remove  
    Database, 304  
db\_view\_setquery  
    Database, 304  
db\_vquery  
    Database, 305  
defer  
    Threads, 719  
defer\_low  
    Threads, 720  
Dictionary, 305  
    dictionary\_appendatom, 311  
    dictionary\_appendatomarray, 312  
    dictionary\_appendatoms, 312  
    dictionary\_appenddictionary, 313  
    dictionary\_appendfloat, 313  
    dictionary\_appendlong, 314  
    dictionary\_appendobject, 314  
    dictionary\_appendstring, 315  
    dictionary\_appendsym, 315  
    dictionary\_chuckentry, 316  
    dictionary\_clear, 316  
    dictionary\_copyatoms, 317  
    dictionary\_copydefatoms, 317  
    dictionary\_copyentries, 318  
    dictionary\_copyunique, 319  
    dictionary\_deleteentry, 319  
    dictionary\_dump, 320  
    dictionary\_entry\_getkey, 320  
    dictionary\_entry\_getvalue, 321  
    dictionary\_entry\_getvalues, 321  
    dictionary\_entryisatomarray, 322  
    dictionary\_entryisdictionary, 322  
    dictionary\_entryisstring, 323  
    dictionary\_freekeys, 323  
    dictionary\_funall, 324  
    dictionary\_get\_ex, 324  
    dictionary\_getatom, 325  
    dictionary\_getatomarray, 325

dictionary\_getatoms, 326  
 dictionary\_getatoms\_ext, 326  
 dictionary\_getdefatom, 327  
 dictionary\_getdefatoms, 328  
 dictionary\_getdeffloat, 328  
 dictionary\_getdeflong, 329  
 dictionary\_getdefstring, 330  
 dictionary\_getdefsym, 330  
 dictionary\_getdictionary, 331  
 dictionary\_getentrycount, 331  
 dictionary\_getfloat, 332  
 dictionary\_getkeys, 332  
 dictionary\_getlong, 333  
 dictionary\_getobject, 333  
 dictionary\_getstring, 334  
 dictionary\_getsym, 334  
 dictionary\_hasentry, 335  
 dictionary\_new, 335  
 dictionary\_read, 336  
 dictionary\_read\_yaml, 336  
 dictionary\_sprintf, 337  
 dictionary\_transaction\_lock, 337  
 dictionary\_transaction\_unlock, 338  
 dictionary\_write, 338  
 dictionary\_write\_yaml, 339  
 postdictionary, 339  
 Dictionary Passing API, 393  
 dictobj\_atom\_safety, 397  
 dictobj\_atom\_safety\_flags, 397  
 dictobj\_dictionaryfromatoms, 397  
 dictobj\_dictionaryfromatoms\_extended, 398  
 dictobj\_dictionaryfromstring, 399  
 dictobj\_dictionarytoatoms, 399  
 dictobj\_findregistered\_clone, 400  
 dictobj\_findregistered\_retain, 400  
 dictobj\_jsonfromstring, 401  
 dictobj\_key\_parse, 401  
 dictobj\_namefromptr, 402  
 dictobj\_outlet\_atoms, 402  
 dictobj\_register, 403  
 dictobj\_release, 403  
 dictobj\_unregister, 404  
 dictobj\_validate, 404  
 dictionary\_appendatom  
     Dictionary, 311  
 dictionary\_appendatomarray  
     Dictionary, 312  
 dictionary\_appendatoms  
     Dictionary, 312  
 dictionary\_appenddictionary  
     Dictionary, 313  
 dictionary\_appendfloat  
     Dictionary, 313  
 dictionary\_appendlong

Dictionary, 314  
 dictionary\_appendobject  
     Dictionary, 314  
 dictionary\_appendstring  
     Dictionary, 315  
 dictionary\_appendsym  
     Dictionary, 315  
 dictionary\_chuckentry  
     Dictionary, 316  
 dictionary\_clear  
     Dictionary, 316  
 dictionary\_copyatoms  
     Dictionary, 317  
 dictionary\_copydefatoms  
     Dictionary, 317  
 dictionary\_copyentries  
     Dictionary, 318  
 dictionary\_copyunique  
     Dictionary, 319  
 dictionary\_deleteentry  
     Dictionary, 319  
 dictionary\_dump  
     Dictionary, 320  
 dictionary\_entry\_getkey  
     Dictionary, 320  
 dictionary\_entry\_getvalue  
     Dictionary, 321  
 dictionary\_entry\_getvalues  
     Dictionary, 321  
 dictionary\_entryisatomarray  
     Dictionary, 322  
 dictionary\_entryisdictionary  
     Dictionary, 322  
 dictionary\_entryisstring  
     Dictionary, 323  
 dictionary\_freekeys  
     Dictionary, 323  
 dictionary\_funall  
     Dictionary, 324  
 dictionary\_get\_ex  
     Dictionary, 324  
 dictionary\_getatom  
     Dictionary, 325  
 dictionary\_getatomarray  
     Dictionary, 325  
 dictionary\_getatoms  
     Dictionary, 326  
 dictionary\_getatoms\_ext  
     Dictionary, 326  
 dictionary\_getdefatom  
     Dictionary, 327  
 dictionary\_getdefatoms  
     Dictionary, 328  
 dictionary\_getdeffloat

Dictionary, 328  
dictionary\_getdeflong  
    Dictionary, 329  
dictionary\_getdefstring  
    Dictionary, 330  
dictionary\_getdefsym  
    Dictionary, 330  
dictionary\_getdictionary  
    Dictionary, 331  
dictionary\_getentrycount  
    Dictionary, 331  
dictionary\_getfloat  
    Dictionary, 332  
dictionary\_getkeys  
    Dictionary, 332  
dictionary\_getlong  
    Dictionary, 333  
dictionary\_getobject  
    Dictionary, 333  
dictionary\_getstring  
    Dictionary, 334  
dictionary\_getsym  
    Dictionary, 334  
dictionary\_hasentry  
    Dictionary, 335  
dictionary\_new  
    Dictionary, 335  
dictionary\_read  
    Dictionary, 336  
dictionary\_read\_yaml  
    Dictionary, 336  
dictionary\_sprintf  
    Dictionary, 337  
dictionary\_transaction\_lock  
    Dictionary, 337  
dictionary\_transaction\_unlock  
    Dictionary, 338  
dictionary\_write  
    Dictionary, 338  
dictionary\_write\_yaml  
    Dictionary, 339  
dictobj\_atom\_safety  
    Dictionary Passing API, 397  
dictobj\_atom\_safety\_flags  
    Dictionary Passing API, 397  
dictobj\_dictionaryfromatoms  
    Dictionary Passing API, 397  
dictobj\_dictionaryfromatoms\_extended  
    Dictionary Passing API, 398  
dictobj\_dictionaryfromstring  
    Dictionary Passing API, 399  
dictobj\_dictionarytoatoms  
    Dictionary Passing API, 399  
dictobj\_findregistered\_clone  
    Dictionary Passing API, 400  
dictobj\_findregistered\_retain  
    Dictionary Passing API, 400  
dictobj\_jsonfromstring  
    Dictionary Passing API, 401  
dictobj\_key\_parse  
    Dictionary Passing API, 401  
dictobj\_namefromptr  
    Dictionary Passing API, 402  
dictobj\_outlet\_atoms  
    Dictionary Passing API, 402  
dictobj\_register  
    Dictionary Passing API, 403  
dictobj\_release  
    Dictionary Passing API, 403  
dictobj\_unregister  
    Dictionary Passing API, 404  
dictobj\_validate  
    Dictionary Passing API, 404  
disposhandle  
    Memory Management, 489  
dsp\_add  
    MSP, 552  
dsp\_addv  
    MSP, 553  
  
e\_max\_atom\_gettext\_flags  
    Atoms, 409  
e\_max\_atomtypes  
    Atoms, 410  
e\_max\_attrflags  
    Attributes, 215  
e\_max\_class\_flags  
    Classes, 249  
e\_max\_datastore\_flags  
    Data Storage, 280  
e\_max\_datflags  
    Systime API, 693  
e\_max\_errorcodes  
    Miscellaneous, 507  
e\_max\_expr\_types  
    Extending expr, 526  
e\_max\_fileinfo\_flags  
    Files and Folders, 450  
e\_max\_openfile\_permissions  
    Files and Folders, 450  
e\_max\_path\_folder\_flags  
    Files and Folders, 450  
e\_max\_path\_styles  
    Files and Folders, 452  
e\_max\_path\_types  
    Files and Folders, 452  
e\_max\_systhread\_mutex\_flags  
    Threads, 719

e\_max\_wind\_advise\_result  
    Miscellaneous, 507

eAltKey  
    Mouse and Keyboard, 544

eAutoRepeat  
    Mouse and Keyboard, 544

eCapsLock  
    Mouse and Keyboard, 544

eCommandKey  
    Mouse and Keyboard, 544

eControlKey  
    Mouse and Keyboard, 544

egetfn  
    Old-Style Classes, 263

eLeftButton  
    Mouse and Keyboard, 544

eMiddleButton  
    Mouse and Keyboard, 544

ePopupMenu  
    Mouse and Keyboard, 544

eRightButton  
    Mouse and Keyboard, 544

error  
    Console, 517

error\_subscribe  
    Miscellaneous, 508

error\_sym  
    Miscellaneous, 508

error\_unsubscribe  
    Miscellaneous, 508

eShiftKey  
    Mouse and Keyboard, 544

ET\_FI  
    Extending expr, 527

ET\_FLT  
    Extending expr, 527

ET\_FUNC  
    Extending expr, 527

ET\_II  
    Extending expr, 527

ET\_INT  
    Extending expr, 527

ET\_LB  
    Extending expr, 527

ET\_LP  
    Extending expr, 527

ET\_OP  
    Extending expr, 527

ET\_SI  
    Extending expr, 527

ET\_STR  
    Extending expr, 527

ET\_SYM  
    Extending expr, 527

ET\_TBL  
    Extending expr, 527

ET\_VSYM  
    Extending expr, 527

expr\_eval  
    Extending expr, 527

expr\_new  
    Extending expr, 528

ext\_main  
    Classes, 257

Extending expr, 525  
    e\_max\_expr\_types, 526

ET\_FI, 527

ET\_FLT, 527

ET\_FUNC, 527

ET\_II, 527

ET\_INT, 527

ET\_LB, 527

ET\_LP, 527

ET\_OP, 527

ET\_SI, 527

ET\_STR, 527

ET\_SYM, 527

ET\_TBL, 527

ET\_VSYM, 527

expr\_eval, 527

expr\_new, 528

fileload  
    Loading Max Files, 535

Files and Folders, 444  
    e\_max\_fileinfo\_flags, 450  
    e\_max\_openfile\_permissions, 450  
    e\_max\_path\_folder\_flags, 450  
    e\_max\_path\_styles, 452  
    e\_max\_path\_types, 452  
    fileusage\_addfile, 454  
    fileusage\_addfolder, 454  
    fileusage\_addpackage, 454  
    filewatcher\_new, 455  
    locatefile, 456  
    locatefile\_extended, 456  
    locate filetype, 457  
    open\_dialog, 458  
    open\_promptset, 459  
    path\_absolutepath, 459  
    path\_closefolder, 460  
    path\_createsysfile, 460  
    path\_exists, 461  
    path\_fileinfo, 461  
    PATH\_FILEINFO\_ALIAS, 450  
    PATH\_FILEINFO\_FOLDER, 450  
    PATH\_FILEINFO\_PACKAGE, 450  
    PATH\_FOLDER\_SNIFF, 452

path\_foldernextfile, 462  
path\_from pathname, 463  
path\_getapppath, 463  
path\_getdefault, 463  
path\_getfilemoddate, 464  
path\_getmoddate, 464  
path\_nameconform, 465  
PATH\_NOALIASRESOLUTION, 452  
path\_openfolder, 465  
path\_opensysfile, 466  
PATH\_READ\_PERM, 450  
PATH\_REPORTPACKAGEASFOLDER, 452  
path\_resolvefile, 466  
PATH\_RW\_PERM, 450  
path\_setdefault, 467  
PATH\_STYLE\_COLON, 452  
PATH\_STYLE\_MAX, 452  
PATH\_STYLE\_NATIVE, 452  
PATH\_STYLE\_NATIVE\_WIN, 452  
PATH\_STYLE\_SLASH, 452  
path\_toabsolutestemppath, 467  
path\_topathname, 468  
path\_topotentialname, 468  
PATH\_TYPE\_ABSOLUTE, 453  
PATH\_TYPE\_BOOT, 453  
PATH\_TYPE\_C74, 453  
PATH\_TYPE\_DESKTOP, 453  
PATH\_TYPE\_IGNORE, 452  
PATH\_TYPE\_MAXDB, 453  
PATH\_TYPE\_PATH, 453  
PATH\_TYPE\_PLUGIN, 453  
PATH\_TYPE\_RELATIVE, 453  
PATH\_TYPE\_TEMPFOLDER, 453  
PATH\_TYPE\_TILDE, 453  
PATH\_TYPE\_USERMAX, 453  
PATH\_WRITE\_PERM, 450  
saveas\_dialog, 469  
saveas\_promptset, 470  
saveasdialog\_extended, 470  
SYSFILE\_ATMARK, 453  
sysfile\_close, 471  
SYSFILE\_FROMLEOF, 453  
SYSFILE\_FROMMARK, 453  
SYSFILE\_FROMSTART, 453  
sysfile\_geteof, 472  
sysfile\_getpos, 472  
sysfile\_openhandle, 473  
sysfile\_openptrsize, 473  
sysfile\_read, 474  
sysfile\_readtextfile, 474  
sysfile\_readtohandle, 475  
sysfile\_readtoptr, 475  
sysfile\_seteof, 476  
sysfile\_setpos, 476  
sysfile\_spoolcopy, 477  
sysfile\_write, 477  
sysfile\_writetextfile, 478  
t\_filehandle, 449  
t\_sysfile\_pos\_mode, 453  
t\_sysfile\_text\_flags, 453  
TEXT\_ENCODING\_USE\_FILE, 453  
TEXT\_LB\_MAC, 453  
TEXT\_LB\_NATIVE, 453  
TEXT\_LB\_PC, 453  
TEXT\_LB\_UNIX, 453  
TEXT\_NULL\_TERMINATE, 453  
fileusage\_addfile  
    Files and Folders, 454  
fileusage\_addfolder  
    Files and Folders, 454  
fileusage\_addpackage  
    Files and Folders, 454  
filewatcher\_new  
    Files and Folders, 455  
FILL\_ATTR\_SAVE  
    Styles, 810  
floatin  
    Inlets and Outlets, 270  
floatout  
    Inlets and Outlets, 270  
freebytes  
    Memory Management, 490  
freebytes16  
    Memory Management, 490  
freeobject  
    Old-Style Classes, 263  
gensym  
    Symbols, 443  
gensym\_tr  
    Symbols, 444  
getbytes  
    Memory Management, 491  
getbytes16  
    Memory Management, 491  
getfn  
    Old-Style Classes, 264  
gettime  
    Clocks, 681  
gettime\_forobject  
    Clocks, 682  
globalsymbol\_bind  
    Miscellaneous, 509  
globalsymbol\_dereference  
    Miscellaneous, 509  
globalsymbol\_reference  
    Miscellaneous, 510  
globalsymbol\_unbind

Miscellaneous, 510  
**growhandle**  
 Memory Management, 492

**Hash Table**, 340  
 hashtab\_chuck, 343  
 hashtab\_chuckkey, 343  
 hashtab\_clear, 344  
 hashtab\_delete, 344  
 hashtab\_findfirst, 345  
 hashtab\_flags, 345  
 hashtab\_funall, 346  
 hashtab\_getflags, 346  
 hashtab\_getkeyflags, 347  
 hashtab\_getkeys, 347  
 hashtab\_getsize, 348  
 hashtab\_keyflags, 348  
 hashtab\_lookup, 349  
 hashtab\_lookupflags, 350  
 hashtab\_lookuplong, 350  
 hashtab\_lookupsym, 351  
 hashtab\_methodall, 351  
 hashtab\_new, 352  
 hashtab\_print, 352  
 hashtab\_READONLY, 353  
 hashtab\_store, 353  
 hashtab\_store\_safe, 354  
 hashtab\_storeflags, 354  
 hashtab\_storelong, 355  
 hashtab\_storesym, 355

hashtab\_chuck  
 Hash Table, 343  
 hashtab\_chuckkey  
 Hash Table, 343  
 hashtab\_clear  
 Hash Table, 344  
 hashtab\_delete  
 Hash Table, 344  
 hashtab\_findfirst  
 Hash Table, 345  
 hashtab\_flags  
 Hash Table, 345  
 hashtab\_funall  
 Hash Table, 346  
 hashtab\_getflags  
 Hash Table, 346  
 hashtab\_getkeyflags  
 Hash Table, 347  
 hashtab\_getkeys  
 Hash Table, 347  
 hashtab\_getsize  
 Hash Table, 348  
 hashtab\_keyflags  
 Hash Table, 348

hashtab\_lookup  
 Hash Table, 349  
 hashtab\_lookupflags  
 Hash Table, 350  
 hashtab\_lookuplong  
 Hash Table, 350  
 hashtab\_lookupsym  
 Hash Table, 351  
 hashtab\_methodall  
 Hash Table, 351  
 hashtab\_new  
 Hash Table, 352  
 hashtab\_print  
 Hash Table, 352  
 hashtab\_READONLY  
 Hash Table, 353  
 hashtab\_store  
 Hash Table, 353  
 hashtab\_store\_safe  
 Hash Table, 354  
 hashtab\_storeflags  
 Hash Table, 354  
 hashtab\_storelong  
 Hash Table, 355  
 hashtab\_storesym  
 Hash Table, 355

**HitBorder**  
 jbox, 633

**HitBox**  
 jbox, 633

**HitGrowBox**  
 jbox, 633

**HitInlet**  
 jbox, 633

**HitLine**  
 jbox, 633

**HitLineLocked**  
 jbox, 633

**HitNothing**  
 jbox, 633

**HitOutlet**  
 jbox, 633

**HitTestResult**  
 jbox, 633

**Index Map**, 356  
 indexmap\_append, 357  
 indexmap\_clear, 358  
 indexmap\_datafromindex, 358  
 indexmap\_delete, 358  
 indexmap\_delete\_index, 359  
 indexmap\_delete\_index\_multi, 359  
 indexmap\_delete\_multi, 360  
 indexmap\_getsize, 360

indexmap\_indexfromdata, 360  
indexmap\_move, 361  
indexmap\_new, 361  
indexmap\_sort, 362  
indexmap\_append  
    Index Map, 357  
indexmap\_clear  
    Index Map, 358  
indexmap\_datafromindex  
    Index Map, 358  
indexmap\_delete  
    Index Map, 358  
indexmap\_delete\_index\_multi  
    Index Map, 359  
indexmap\_delete\_index\_multi  
    Index Map, 359  
indexmap\_getsize  
    Index Map, 360  
indexmap\_indexfromdata  
    Index Map, 360  
indexmap\_move  
    Index Map, 361  
indexmap\_new  
    Index Map, 361  
indexmap\_sort  
    Index Map, 362  
inlet\_new  
    Inlets and Outlets, 271  
Inlets and Outlets, 268  
    bangout, 269  
    floatin, 270  
    floatout, 270  
    inlet\_new, 271  
    intin, 271  
    intout, 272  
    listout, 272  
    outlet\_anything, 273  
    outlet\_bang, 274  
    outlet\_float, 274  
    outlet\_int, 274  
    outlet\_list, 275  
    outlet\_new, 276  
    proxy\_getinlet, 276  
    proxy\_new, 277  
INRANGE  
    Miscellaneous, 505  
intin  
    Inlets and Outlets, 271  
intload  
    Loading Max Files, 535  
intout  
    Inlets and Outlets, 272  
isr  
    Threads, 721  
ITM Time Objects, 696  
    class\_time\_addattr, 700  
    itm\_barbeatunitstoticks, 701  
    itm\_dereference, 701  
    itm\_dump, 702  
    itm\_getglobal, 702  
    itm\_getname, 702  
    itm\_getnamed, 702  
    itm\_getresolution, 703  
    itm\_getstate, 703  
    itm\_getticks, 704  
    itm\_gettime, 704  
    itm\_gettimesignature, 705  
    itm\_isunitfixed, 705  
    itm\_mstosamps, 705  
    itm\_mstoticks, 706  
    itm\_pause, 706  
    itm\_reference, 707  
    itm\_resume, 707  
    itm\_sampstoms, 707  
    itm\_setresolution, 708  
    itm\_settimesignature, 708  
    itm\_ticksbarbeatunits, 709  
    itm\_tickstoms, 709  
t\_itm, 699  
t\_timeobject, 716  
time\_calcquantize, 709  
TIME\_FLAGS\_BBUSOURCE, 700  
TIME\_FLAGS\_CHECKSCHEDULE, 700  
TIME\_FLAGS\_EVENTLIST, 700  
TIME\_FLAGS\_FIXED, 700  
TIME\_FLAGS\_FIXEDONLY, 700  
TIME\_FLAGS\_LISTENTICKS, 700  
TIME\_FLAGS\_LOCATION, 700  
TIME\_FLAGS\_LOOKAHEAD, 700  
TIME\_FLAGS\_NOUNITS, 700  
TIME\_FLAGS\_PERMANENT, 700  
TIME\_FLAGS\_POSITIVE, 700  
TIME\_FLAGS\_TICKSONLY, 700  
TIME\_FLAGS\_TRANSPORT, 700  
TIME\_FLAGS\_USECLOCK, 700  
TIME\_FLAGS\_USEQELEM, 700  
time\_getitm, 710  
time\_getms, 710  
time\_getnamed, 711  
time\_getphase, 711  
time\_getticks, 712  
time\_isfixedunit, 712  
time\_listen, 712  
time\_new, 713  
time\_now, 713  
time\_schedule, 714

time\_schedule\_limit, 714  
 time\_setclock, 714  
 time\_setvalue, 715  
 time\_stop, 715  
 time\_tick, 715  
 itm\_barbeatunitstoticks  
     ITM Time Objects, 701  
 itm\_dereference  
     ITM Time Objects, 701  
 itm\_dump  
     ITM Time Objects, 702  
 itm\_getglobal  
     ITM Time Objects, 702  
 itm\_getname  
     ITM Time Objects, 702  
 itm\_getnamed  
     ITM Time Objects, 702  
 itm\_getresolution  
     ITM Time Objects, 703  
 itm\_getstate  
     ITM Time Objects, 703  
 itm\_getticks  
     ITM Time Objects, 704  
 itm\_gettime  
     ITM Time Objects, 704  
 itm\_gettimesignature  
     ITM Time Objects, 705  
 itm\_isunitfixed  
     ITM Time Objects, 705  
 itm\_mstosamps  
     ITM Time Objects, 705  
 itm\_mstoticks  
     ITM Time Objects, 706  
 itm\_pause  
     ITM Time Objects, 706  
 itm\_reference  
     ITM Time Objects, 707  
 itm\_resume  
     ITM Time Objects, 707  
 itm\_sampstoms  
     ITM Time Objects, 707  
 itm\_setresolution  
     ITM Time Objects, 708  
 itm\_settimesignature  
     ITM Time Objects, 708  
 itm\_ticksbarbeatunits  
     ITM Time Objects, 709  
 itm\_tickstoms  
     ITM Time Objects, 709  
  
 jbox, 627  
     HitBorder, 633  
     HitBox, 633  
     HitGrowBox, 633  
  
     HitInlet, 633  
     HitLine, 633  
     HitLineLocked, 633  
     HitNothing, 633  
     HitOutlet, 633  
     HitTestResult, 633  
     JBOX\_FONTFACE\_BOLD, 632  
     JBOX\_FONTFACE\_BOLDITALIC, 632  
     JBOX\_FONTFACE\_ITALIC, 632  
     JBOX\_FONTFACE\_REGULAR, 632  
     jbox\_free, 633  
     jbox\_get\_annotation, 633  
     jbox\_get\_background, 635  
     jbox\_get\_canhilite, 635  
     jbox\_get\_color, 635  
     jbox\_get\_drawfirstin, 636  
     jbox\_get\_drawinlast, 636  
     jbox\_get\_fontname, 637  
     jbox\_get\_fontsize, 637  
     jbox\_get\_growboth, 637  
     jbox\_get\_growy, 638  
     jbox\_get\_hidden, 638  
     jbox\_get\_hint, 638  
     jbox\_get\_hintstring, 639  
     jbox\_get\_id, 639  
     jbox\_get\_ignoreclick, 640  
     jbox\_get\_maxclass, 640  
     jbox\_get\_nextobject, 640  
     jbox\_get\_nogrow, 641  
     jbox\_get\_object, 641  
     jbox\_get\_outline, 641  
     jbox\_get\_patcher, 642  
     jbox\_get\_patching\_position, 642  
     jbox\_get\_patching\_rect, 643  
     jbox\_get\_patching\_size, 643  
     jbox\_get\_presentation, 643  
     jbox\_get\_presentation\_position, 644  
     jbox\_get\_presentation\_rect, 644  
     jbox\_get\_presentation\_size, 645  
     jbox\_get\_prevobject, 645  
     jbox\_get\_rect\_for\_sym, 646  
     jbox\_get\_rect\_for\_view, 646  
     jbox\_get\_textfield, 646  
     jbox\_get\_varname, 647  
     jbox\_new, 647  
     JBOX\_NOINSPECTFIRSTIN, 632  
     jbox\_notify, 648  
     jbox\_ready, 648  
     jbox\_redraw, 650  
     jbox\_set\_annotation, 650  
     jbox\_set\_background, 650  
     jbox\_set\_color, 651  
     jbox\_set\_fontname, 651  
     jbox\_set\_fontsize, 652

jbox\_set\_hidden, 652  
jbox\_set\_hint, 652  
jbox\_set\_hintstring, 653  
jbox\_set\_ignoreclick, 653  
jbox\_set\_outline, 654  
jbox\_set\_patching\_position, 654  
jbox\_set\_patching\_rect, 655  
jbox\_set\_patching\_size, 655  
jbox\_set\_position, 655  
jbox\_set\_presentation, 656  
jbox\_set\_presentation\_position, 656  
jbox\_set\_presentation\_rect, 657  
jbox\_set\_presentation\_size, 657  
jbox\_set\_rect, 658  
jbox\_set\_rect\_for\_sym, 658  
jbox\_set\_rect\_for\_view, 658  
jbox\_set\_size, 659  
jbox\_set\_varname, 659  
jbox\_end\_layer  
    Box Layer, 843  
JBOX\_FONTFACE\_BOLD  
    jbox, 632  
JBOX\_FONTFACE\_BOLDITALIC  
    jbox, 632  
JBOX\_FONTFACE\_ITALIC  
    jbox, 632  
JBOX\_FONTFACE\_REGULAR  
    jbox, 632  
jbox\_free  
    jbox, 633  
jbox\_get\_annotation  
    jbox, 633  
jbox\_get\_background  
    jbox, 635  
jbox\_get\_canhilite  
    jbox, 635  
jbox\_get\_color  
    jbox, 635  
jbox\_get\_drawfirstin  
    jbox, 636  
jbox\_get\_drawinlast  
    jbox, 636  
jbox\_get\_font\_slant  
    JFont, 783  
jbox\_get\_font\_weight  
    JFont, 784  
jbox\_get\_fontname  
    jbox, 637  
jbox\_get\_fontsize  
    jbox, 637  
jbox\_get\_growboth  
    jbox, 637  
jbox\_get\_growy  
    jbox, 638  
jbox\_get\_hidden  
    jbox, 638  
jbox\_get\_hint  
    jbox, 638  
jbox\_get\_hintstring  
    jbox, 639  
jbox\_get\_id  
    jbox, 639  
jbox\_get\_ignoreclick  
    jbox, 640  
jbox\_get\_maxclass  
    jbox, 640  
jbox\_get\_nextobject  
    jbox, 640  
jbox\_get\_nogrow  
    jbox, 641  
jbox\_get\_object  
    jbox, 641  
jbox\_get\_outline  
    jbox, 641  
jbox\_get\_patcher  
    jbox, 642  
jbox\_get\_patching\_position  
    jbox, 642  
jbox\_get\_patching\_rect  
    jbox, 643  
jbox\_get\_patching\_size  
    jbox, 643  
jbox\_get\_presentation  
    jbox, 643  
jbox\_get\_presentation\_position  
    jbox, 644  
jbox\_get\_presentation\_rect  
    jbox, 644  
jbox\_get\_presentation\_size  
    jbox, 645  
jbox\_get\_prevobject  
    jbox, 645  
jbox\_get\_rect\_for\_sym  
    jbox, 646  
jbox\_get\_rect\_for\_view  
    jbox, 646  
jbox\_get\_textfield  
    jbox, 646  
jbox\_get\_varname  
    jbox, 647  
jbox\_invalidate\_layer  
    Box Layer, 843  
jbox\_new  
    jbox, 647  
JBOX\_NOINSPECTFIRSTIN  
    jbox, 632  
jbox\_notify  
    jbox, 648

jbox\_paint\_layer  
     Box Layer, 844

jbox\_ready  
     jbox, 648

jbox\_redraw  
     jbox, 650

jbox\_set\_annotation  
     jbox, 650

jbox\_set\_background  
     jbox, 650

jbox\_set\_color  
     jbox, 651

jbox\_set\_fontname  
     jbox, 651

jbox\_set fontsize  
     jbox, 652

jbox\_set\_hidden  
     jbox, 652

jbox\_set\_hint  
     jbox, 652

jbox\_set\_ignoreclick  
     jbox, 653

jbox\_set\_outline  
     jbox, 654

jbox\_set\_patching\_position  
     jbox, 654

jbox\_set\_patching\_rect  
     jbox, 655

jbox\_set\_patching\_size  
     jbox, 655

jbox\_set\_position  
     jbox, 655

jbox\_set\_presentation  
     jbox, 656

jbox\_set\_presentation\_position  
     jbox, 656

jbox\_set\_presentation\_rect  
     jbox, 657

jbox\_set\_presentation\_size  
     jbox, 657

jbox\_set\_rect  
     jbox, 658

jbox\_set\_rect\_for\_sym  
     jbox, 658

jbox\_set\_rect\_for\_view  
     jbox, 658

jbox\_set\_size  
     jbox, 659

jbox\_set\_varname  
     jbox, 659

jbox\_start\_layer  
     Box Layer, 844

jdataview\_getclient  
     DataView, 846

jdataview\_new  
     DataView, 846

jdataview\_setclient  
     DataView, 847

JFont, 781

- jbox\_get\_font\_slant, 783
- jbox\_get\_font\_weight, 784
- jfont\_create, 784
- jfont\_create\_withstylename, 785
- jfont\_destroy, 785
- jfont\_extents, 786
- jfont\_get\_em\_dimensions, 786
- jfont\_get\_family, 786
- jfont\_get\_font\_size, 787
- jfont\_get\_slant, 787
- jfont\_get\_style, 787
- jfont\_get\_underline, 788
- jfont\_get\_weight, 788
- jfont\_getfontlist, 789
- jfont\_getfontstylenames, 789
- jfont\_isequalto, 790
- jfont\_normalizefontname, 790
- jfont\_reference, 790
- jfont\_set\_family, 791
- jfont\_set\_font\_size, 791
- jfont\_set\_slant, 792
- jfont\_set\_style, 792
- jfont\_set\_underline, 792
- jfont\_set\_weight, 793
- jfont\_text\_measure, 793
- jfont\_text\_measuretext\_wrapped, 793
- JGRAPHICS\_FONT\_SLANT\_ITALIC, 783
- JGRAPHICS\_FONT\_SLANT\_NORMAL, 783
- JGRAPHICS\_FONT\_WEIGHT\_BOLD, 783
- JGRAPHICS\_FONT\_WEIGHT\_NORMAL, 783
- systemfontname, 794
- systemfontname\_bold, 794
- systemfontname\_light, 794
- systemfontsym, 795
- t\_jgraphics\_font\_slant, 783
- t\_jgraphics\_font\_weight, 783

jfont\_create  
     JFont, 784

jfont\_create\_withstylename  
     JFont, 785

jfont\_destroy  
     JFont, 785

jfont\_extents  
     JFont, 786

jfont\_get\_em\_dimensions  
     JFont, 786

jfont\_get\_family

JFont, 786  
jfont\_get\_font\_size  
    JFont, 787  
jfont\_get\_slant  
    JFont, 787  
jfont\_get\_style  
    JFont, 787  
jfont\_get\_underline  
    JFont, 788  
jfont\_get\_weight  
    JFont, 788  
jfont\_getfontlist  
    JFont, 789  
jfont\_getfontstylenames  
    JFont, 789  
jfont\_isequalto  
    JFont, 790  
jfont\_normalizefontname  
    JFont, 790  
jfont\_reference  
    JFont, 790  
jfont\_set\_family  
    JFont, 791  
jfont\_set\_font\_size  
    JFont, 791  
jfont\_set\_slant  
    JFont, 792  
jfont\_set\_style  
    JFont, 792  
jfont\_set\_underline  
    JFont, 792  
jfont\_set\_weight  
    JFont, 793  
jfont\_text\_measure  
    JFont, 793  
jfont\_text\_measuretext\_wrapped  
    JFont, 793  
JGraphics, 736  
    JGRAPHICS\_2PI, 741  
    JGRAPHICS\_3PIOVER2, 741  
    jgraphics\_append\_path, 744  
    jgraphics\_arc, 745  
    jgraphics\_arc\_negative, 745  
    jgraphics\_bubble, 746  
    jgraphics\_clip, 746  
    jgraphics\_close\_path, 747  
    jgraphics\_copy\_path, 747  
    jgraphics\_curve\_to, 747  
    jgraphics\_destroy, 748  
    jgraphics\_device\_to\_user, 748  
    jgraphics\_ellipse, 748  
    JGRAPHICS\_FILEFORMAT\_JPEG, 742  
    JGRAPHICS\_FILEFORMAT\_PNG, 742  
    jgraphics\_font\_extents, 749  
    JGRAPHICS\_FORMAT\_A8, 744  
    JGRAPHICS\_FORMAT\_ARGB32, 744  
    JGRAPHICS\_FORMAT\_RGB24, 744  
    jgraphics\_get\_current\_point, 749  
    jgraphics\_getfiletypes, 750  
    jgraphics\_line\_to, 750  
    jgraphics\_move\_to, 751  
    jgraphics\_new\_path, 751  
    jgraphics\_oval, 752  
    jgraphics\_ovalarc, 752  
    jgraphics\_path\_contains, 753  
    jgraphics\_path\_createstroked, 753  
    jgraphics\_path\_destroy, 753  
    jgraphics\_path\_getlength, 755  
    jgraphics\_path\_getnearestpoint, 755  
    jgraphics\_path\_getpathelems, 756  
    jgraphics\_path\_getpointalongpath, 756  
    jgraphics\_path\_intersectsline, 756  
    jgraphics\_path\_roundcorners, 757  
    JGRAPHICS\_PI, 742  
    JGRAPHICS\_PIOVER2, 742  
    jgraphics\_position\_one\_rect\_near\_another\_rect\_but\_keep\_inside\_a\_th  
        757  
    jgraphics\_rectangle, 758  
    jgraphics\_rectangle\_rounded, 758  
    jgraphics\_rectcontainsrect, 759  
    jgraphics\_rectintersectsrect, 759  
    jgraphics\_reference, 760  
    jgraphics\_rel\_curve\_to, 760  
    jgraphics\_rel\_line\_to, 761  
    jgraphics\_rel\_move\_to, 761  
    jgraphics\_round, 761  
    jgraphics\_select\_font\_face, 762  
    jgraphics\_select\_jfont, 762  
    jgraphics\_set\_font\_size, 762  
    jgraphics\_set\_underline, 763  
    jgraphics\_show\_text, 763  
    jgraphics\_system\_canganitaliastexttotransparentbg,  
        763  
    JGRAPHICS\_TEXT JUSTIFICATION BOTTOM,  
        744  
    JGRAPHICS\_TEXT JUSTIFICATION CENTERED,  
        744  
    JGRAPHICS\_TEXT JUSTIFICATION HCENTERED,  
        744  
    JGRAPHICS\_TEXT JUSTIFICATION HJUSTIFIED,  
        744  
    JGRAPHICS\_TEXT JUSTIFICATION LEFT, 744  
    JGRAPHICS\_TEXT JUSTIFICATION RIGHT, 744  
    JGRAPHICS\_TEXT JUSTIFICATION TOP, 744  
    JGRAPHICS\_TEXT JUSTIFICATION VCENTERED,  
        744  
    jgraphics\_text\_measure, 764  
    jgraphics\_text\_measuretext\_wrapped, 764

jgraphics\_text\_path, 765  
jgraphics\_triangle, 765  
jgraphics\_user\_to\_device, 766  
t\_jgraphics\_fileformat, 742  
t\_jgraphics\_format, 742  
t\_jgraphics\_text\_justification, 744  
JGraphics Matrix Transformations, 795  
jgraphics\_matrix\_init, 796  
jgraphics\_matrix\_init\_identity, 797  
jgraphics\_matrix\_init\_rotate, 797  
jgraphics\_matrix\_init\_scale, 797  
jgraphics\_matrix\_init\_translate, 798  
jgraphics\_matrix\_invert, 798  
jgraphics\_matrix\_multiply, 799  
jgraphics\_matrix\_rotate, 799  
jgraphics\_matrix\_scale, 799  
jgraphics\_matrix\_transform\_point, 800  
jgraphics\_matrix\_translate, 800  
JGRAPHICS\_2PI  
JGraphics, 741  
JGRAPHICS\_3PIOVER2  
JGraphics, 741  
jgraphics\_append\_path  
JGraphics, 744  
jgraphics\_arc  
JGraphics, 745  
jgraphics\_arc\_negative  
JGraphics, 745  
jgraphics\_attr\_fillrect  
Styles, 812  
jgraphics\_attr\_setfill  
Styles, 813  
jgraphics\_bubble  
JGraphics, 746  
jgraphics\_clip  
JGraphics, 746  
jgraphics\_close\_path  
JGraphics, 747  
jgraphics\_copy\_path  
JGraphics, 747  
jgraphics\_create  
JSurface, 768  
jgraphics\_curve\_to  
JGraphics, 747  
jgraphics\_destroy  
JGraphics, 748  
jgraphics\_device\_to\_user  
JGraphics, 748  
jgraphics\_ellipse  
JGraphics, 748  
JGRAPHICS\_FILEFORMAT\_JPEG  
JGraphics, 742  
JGRAPHICS\_FILEFORMAT\_PNG  
JGraphics, 742  
jgraphics\_font\_extents  
JGraphics, 749  
JGRAPHICS\_FONT\_SLANT\_ITALIC  
JFont, 783  
JGRAPHICS\_FONT\_SLANT\_NORMAL  
JFont, 783  
JGRAPHICS\_FONT\_WEIGHT\_BOLD  
JFont, 783  
JGRAPHICS\_FONT\_WEIGHT\_NORMAL  
JFont, 783  
JGRAPHICS\_FORMAT\_A8  
JGraphics, 744  
JGRAPHICS\_FORMAT\_ARGB32  
JGraphics, 744  
JGRAPHICS\_FORMAT\_RGB24  
JGraphics, 744  
jgraphics\_get\_current\_point  
JGraphics, 749  
jgraphics\_get\_resource\_data  
JSurface, 768  
jgraphics\_getfiletypes  
JGraphics, 750  
jgraphics\_image\_surface\_clear  
JSurface, 769  
jgraphics\_image\_surface\_create  
JSurface, 769  
jgraphics\_image\_surface\_create\_for\_data  
JSurface, 770  
jgraphics\_image\_surface\_create\_from\_file  
JSurface, 770  
jgraphics\_image\_surface\_create\_from\_filedata  
JSurface, 771  
jgraphics\_image\_surface\_create\_from\_resource  
JSurface, 771  
jgraphics\_image\_surface\_create\_referenced  
JSurface, 772  
jgraphics\_image\_surface\_draw  
JSurface, 772  
jgraphics\_image\_surface\_draw\_fast  
JSurface, 773  
jgraphics\_image\_surface\_get\_height  
JSurface, 773  
jgraphics\_image\_surface\_get\_pixel  
JSurface, 774  
jgraphics\_image\_surface\_get\_width  
JSurface, 774  
jgraphics\_image\_surface\_scroll  
JSurface, 775  
jgraphics\_image\_surface\_set\_pixel  
JSurface, 775  
jgraphics\_image\_surface\_writejpeg  
JSurface, 776  
jgraphics\_image\_surface\_writepng  
JSurface, 776

jgraphics\_line\_to  
JGraphics, 750

jgraphics\_matrix\_init  
JGraphics Matrix Transformations, 796

jgraphics\_matrix\_init\_identity  
JGraphics Matrix Transformations, 797

jgraphics\_matrix\_init\_rotate  
JGraphics Matrix Transformations, 797

jgraphics\_matrix\_init\_scale  
JGraphics Matrix Transformations, 797

jgraphics\_matrix\_init\_translate  
JGraphics Matrix Transformations, 798

jgraphics\_matrix\_invert  
JGraphics Matrix Transformations, 798

jgraphics\_matrix\_multiply  
JGraphics Matrix Transformations, 799

jgraphics\_matrix\_rotate  
JGraphics Matrix Transformations, 799

jgraphics\_matrix\_scale  
JGraphics Matrix Transformations, 799

jgraphics\_matrix\_transform\_point  
JGraphics Matrix Transformations, 800

jgraphics\_matrix\_translate  
JGraphics Matrix Transformations, 800

jgraphics\_move\_to  
JGraphics, 751

jgraphics\_new\_path  
JGraphics, 751

jgraphics\_oval  
JGraphics, 752

jgraphics\_ovalarc  
JGraphics, 752

jgraphics\_path\_contains  
JGraphics, 753

jgraphics\_path\_createstroked  
JGraphics, 753

jgraphics\_path\_destroy  
JGraphics, 753

jgraphics\_path\_getlength  
JGraphics, 755

jgraphics\_path\_getnearestpoint  
JGraphics, 755

jgraphics\_path\_getpathelems  
JGraphics, 756

jgraphics\_path\_getpointalongpath  
JGraphics, 756

jgraphics\_path\_intersectsline  
JGraphics, 756

jgraphics\_path\_roundcorners  
JGraphics, 757

JGRAPHICS\_PI  
JGraphics, 742

JGRAPHICS\_PIOVER2  
JGraphics, 742

jgraphics\_position\_one\_rect\_near\_another\_rect\_but\_keep\_inside\_a\_third\_rect  
JGraphics, 757

jgraphics\_rectangle  
JGraphics, 758

jgraphics\_rectangle\_rounded  
JGraphics, 758

jgraphics\_rectcontainsrect  
JGraphics, 759

jgraphics\_rectintersectsrect  
JGraphics, 759

jgraphics\_reference  
JGraphics, 760

jgraphics\_rel\_curve\_to  
JGraphics, 760

jgraphics\_rel\_line\_to  
JGraphics, 761

jgraphics\_rel\_move\_to  
JGraphics, 761

jgraphics\_round  
JGraphics, 761

jgraphics\_select\_font\_face  
JGraphics, 762

jgraphics\_select\_jfont  
JGraphics, 762

jgraphics\_set\_font\_size  
JGraphics, 762

jgraphics\_set\_underline  
JGraphics, 763

jgraphics\_show\_text  
JGraphics, 763

jgraphics\_surface\_destroy  
JSurface, 776

jgraphics\_surface\_reference  
JSurface, 777

jgraphics\_system\_canantialiastattotransparentbg  
JGraphics, 763

JGRAPHICS\_TEXT\_JUSTIFICATION\_BOTTOM  
JGraphics, 744

JGRAPHICS\_TEXT\_JUSTIFICATION\_CENTERED  
JGraphics, 744

JGRAPHICS\_TEXT\_JUSTIFICATION\_HCENTERED  
JGraphics, 744

JGRAPHICS\_TEXT\_JUSTIFICATION\_HJUSTIFIED  
JGraphics, 744

JGRAPHICS\_TEXT\_JUSTIFICATION\_LEFT  
JGraphics, 744

JGRAPHICS\_TEXT\_JUSTIFICATION\_RIGHT  
JGraphics, 744

JGRAPHICS\_TEXT\_JUSTIFICATION\_TOP  
JGraphics, 744

JGRAPHICS\_TEXT\_JUSTIFICATION\_VCENTERED  
JGraphics, 744

jgraphics\_text\_measure  
JGraphics, 764

jgraphics\_text\_measuretext\_wrapped  
     JGraphics, 764

jgraphics\_text\_path  
     JGraphics, 765

JGRAPHICS\_TEXTLAYOUT\_NOWRAP  
     TextLayout, 829

JGRAPHICS\_TEXTLAYOUT\_USEELLIPSIS  
     TextLayout, 829

jgraphics\_triangle  
     JGraphics, 765

jgraphics\_user\_to\_device  
     JGraphics, 766

jgraphics\_write\_image\_surface\_to\_filedata  
     JSurface, 777

jit\_atom\_arg\_getdouble  
     Atom Module, 852

jit\_atom\_arg\_getfloat  
     Atom Module, 853

jit\_atom\_arg\_getlong  
     Atom Module, 854

jit\_atom\_arg\_getsym  
     Atom Module, 855

jit\_atom\_getcharfix  
     Atom Module, 856

jit\_atom\_getfloat  
     Atom Module, 856

jit\_atom\_getlong  
     Atom Module, 857

jit\_atom\_getobj  
     Atom Module, 858

jit\_atom\_getsym  
     Atom Module, 859

jit\_atom\_setfloat  
     Atom Module, 859

jit\_atom\_setlong  
     Atom Module, 860

jit\_atom\_setobj  
     Atom Module, 861

jit\_atom\_setsym  
     Atom Module, 862

jit\_attr\_canget  
     Attribute Module, 865

jit\_attr\_canset  
     Attribute Module, 865

jit\_attr\_filter\_clip\_new  
     Attribute Module, 866

jit\_attr\_filter\_proc\_new  
     Attribute Module, 866

jit\_attr\_filterget  
     Attribute Module, 867

jit\_attr\_filterset  
     Attribute Module, 868

jit\_attr\_get  
     Attribute Module, 868

    jit\_attr\_getchar\_array  
         Attribute Module, 869

    jit\_attr\_getdouble\_array  
         Attribute Module, 870

    jit\_attr\_getfloat  
         Attribute Module, 871

    jit\_attr\_getfloat\_array  
         Attribute Module, 872

    jit\_attr\_getlong  
         Attribute Module, 873

    jit\_attr\_getlong\_array  
         Attribute Module, 874

    jit\_attr\_getmethod  
         Attribute Module, 875

    jit\_attr\_getname  
         Attribute Module, 875

    jit\_attr\_getsym  
         Attribute Module, 876

    jit\_attr\_getsym\_array  
         Attribute Module, 877

    jit\_attr\_gettime  
         Attribute Module, 878

    jit\_attr\_offset\_array\_new  
         Attribute Module, 878

    jit\_attr\_offset\_new  
         Attribute Module, 879

    jit\_attr\_set  
         Attribute Module, 880

    jit\_attr\_setchar\_array  
         Attribute Module, 881

    jit\_attr\_setdouble\_array  
         Attribute Module, 882

    jit\_attr\_setfloat  
         Attribute Module, 883

    jit\_attr\_setfloat\_array  
         Attribute Module, 884

    jit\_attr\_setlong  
         Attribute Module, 885

    jit\_attr\_setlong\_array  
         Attribute Module, 886

    jit\_attr\_setsym  
         Attribute Module, 887

    jit\_attr\_setsym\_array  
         Attribute Module, 888

    jit\_attr\_symcompare  
         Attribute Module, 889

    jit\_attr\_usercanget  
         Attribute Module, 890

    jit\_attr\_usercanset  
         Attribute Module, 890

    jit\_attribute\_new  
         Attribute Module, 891

    jit\_bin\_read\_chunk\_info  
         Binary Module, 892

jit\_bin\_read\_header  
    Binary Module, 893  
jit\_bin\_read\_matrix  
    Binary Module, 894  
jit\_bin\_write\_header  
    Binary Module, 895  
jit\_bin\_write\_matrix  
    Binary Module, 896  
jit\_class\_adornment  
    Class Module, 899  
jit\_class\_addattr  
    Class Module, 899  
jit\_class\_addinterface  
    Class Module, 900  
jit\_class\_addmethod  
    Class Module, 901  
jit\_class\_addtypedwrapper  
    Class Module, 902  
jit\_class\_adornment\_get  
    Class Module, 903  
jit\_class\_attr\_get  
    Class Module, 903  
jit\_class\_findbyname  
    Class Module, 904  
jit\_class\_free  
    Class Module, 904  
jit\_class\_mess  
    Class Module, 905  
jit\_class\_method  
    Class Module, 906  
jit\_class\_method\_addargsafe  
    Class Module, 906  
jit\_class\_method\_argsafe\_get  
    Class Module, 907  
jit\_class\_nameget  
    Class Module, 908  
jit\_class\_new  
    Class Module, 908  
jit\_class\_register  
    Class Module, 909  
jit\_class\_symcompare  
    Class Module, 910  
jit\_class\_typedwrapper\_get  
    Class Module, 911  
jit\_copy\_bytes  
    Memory Module, 1028  
jit\_disposeptr  
    Memory Module, 1029  
jit\_err\_from\_max\_err  
    Miscellaneous Utility Module, 931  
jit\_error\_code  
    Miscellaneous Utility Module, 931  
jit\_error\_sym  
    Miscellaneous Utility Module, 932  
jit\_freebytes  
    Memory Module, 1029  
jit\_freemem  
    Memory Module, 1030  
jit\_getbytes  
    Memory Module, 1030  
jit\_global\_critical\_enter  
    Miscellaneous Utility Module, 933  
jit\_global\_critical\_exit  
    Miscellaneous Utility Module, 933  
jit\_handle\_free  
    Memory Module, 1031  
jit\_handle\_lock  
    Memory Module, 1032  
jit\_handle\_new  
    Memory Module, 1033  
jit\_handle\_size\_get  
    Memory Module, 1034  
jit\_handle\_size\_set  
    Memory Module, 1034  
jit\_linklist\_append  
    Linked List Module, 937  
jit\_linklist\_chuck  
    Linked List Module, 938  
jit\_linklist\_chuckindex  
    Linked List Module, 939  
jit\_linklist\_clear  
    Linked List Module, 940  
jit\_linklist\_deleteindex  
    Linked List Module, 941  
jit\_linklist\_findall  
    Linked List Module, 942  
jit\_linklist\_findcount  
    Linked List Module, 943  
jit\_linklist\_findfirst  
    Linked List Module, 944  
jit\_linklist\_free  
    Matrix Module, 980  
jit\_linklist\_getindex  
    Linked List Module, 945  
jit\_linklist\_getsize  
    Linked List Module, 946  
jit\_linklist\_insertindex  
    Linked List Module, 947  
jit\_linklist\_makearray  
    Linked List Module, 947  
jit\_linklist\_methodall  
    Linked List Module, 948  
jit\_linklist\_methodindex  
    Linked List Module, 949  
jit\_linklist\_new  
    Linked List Module, 950  
jit\_linklist\_objptr2index  
    Linked List Module, 950

jit\_linklist\_reverse  
    Linked List Module, 951  
jit\_linklist\_rotate  
    Linked List Module, 952  
jit\_linklist\_shuffle  
    Linked List Module, 953  
jit\_linklist\_sort  
    Linked List Module, 953  
jit\_linklist\_swap  
    Linked List Module, 954  
jit\_math\_acos  
    Math Module, 957  
jit\_math\_acosh  
    Math Module, 958  
jit\_math\_asin  
    Math Module, 958  
jit\_math\_asinh  
    Math Module, 958  
jit\_math\_atan  
    Math Module, 959  
jit\_math\_atan2  
    Math Module, 959  
jit\_math\_atanh  
    Math Module, 960  
jit\_math\_ceil  
    Math Module, 960  
jit\_math\_cos  
    Math Module, 960  
jit\_math\_cosh  
    Math Module, 961  
jit\_math\_exp  
    Math Module, 961  
jit\_math\_exp2  
    Math Module, 961  
jit\_math\_expm1  
    Math Module, 962  
jit\_math\_fast acos  
    Math Module, 962  
jit\_math\_fast\_asin  
    Math Module, 963  
jit\_math\_fast\_atan  
    Math Module, 964  
jit\_math\_fast\_cos  
    Math Module, 964  
jit\_math\_fast\_invsqrt  
    Math Module, 965  
jit\_math\_fast\_sin  
    Math Module, 965  
jit\_math\_fast\_sqrt  
    Math Module, 965  
jit\_math\_fast\_tan  
    Math Module, 967  
jit\_math\_floor  
    Math Module, 967  
jit\_math\_fmod  
    Math Module, 967  
jit\_math\_fold  
    Math Module, 968  
jit\_math\_hypot  
    Math Module, 968  
jit\_math\_is\_finite  
    Math Module, 969  
jit\_math\_is\_nan  
    Math Module, 969  
jit\_math\_is\_poweroftwo  
    Math Module, 970  
jit\_math\_is\_valid  
    Math Module, 970  
jit\_math\_j1  
    Math Module, 971  
jit\_math\_j1\_0  
    Math Module, 972  
jit\_math\_log  
    Math Module, 972  
jit\_math\_log10  
    Math Module, 973  
jit\_math\_log2  
    Math Module, 973  
jit\_math\_p1  
    Math Module, 974  
jit\_math\_pow  
    Math Module, 974  
jit\_math\_q1  
    Math Module, 974  
jit\_math\_round  
    Math Module, 975  
jit\_math\_roundup\_poweroftwo  
    Math Module, 975  
jit\_math\_sin  
    Math Module, 976  
jit\_math\_sinh  
    Math Module, 976  
jit\_math\_sqrt  
    Math Module, 976  
jit\_math\_tan  
    Math Module, 977  
jit\_math\_tanh  
    Math Module, 977  
jit\_math\_trunc  
    Math Module, 978  
jit\_math\_wrap  
    Math Module, 978  
jit\_matrix\_clear  
    Matrix Module, 981  
jit\_matrix\_data  
    Matrix Module, 981  
jit\_matrix\_exprfill  
    Matrix Module, 982

jit\_matrix\_fillplane  
    Matrix Module, 983  
jit\_matrix\_free  
    Matrix Module, 984  
jit\_matrix\_freedata  
    Matrix Module, 985  
jit\_matrix\_frommatrix  
    Matrix Module, 986  
jit\_matrix\_getcell  
    Matrix Module, 986  
jit\_matrix\_getdata  
    Matrix Module, 988  
jit\_matrix\_getinfo  
    Matrix Module, 988  
jit\_matrix\_info\_default  
    Matrix Module, 989  
jit\_matrix\_jit\_gl\_texture  
    Matrix Module, 990  
jit\_matrix\_new  
    Matrix Module, 990  
jit\_matrix\_newcopy  
    Matrix Module, 991  
jit\_matrix\_op  
    Matrix Module, 992  
jit\_matrix\_setall  
    Matrix Module, 993  
jit\_matrix\_setcell  
    Matrix Module, 995  
jit\_matrix\_setcell1d  
    Matrix Module, 996  
jit\_matrix\_setcell2d  
    Matrix Module, 997  
jit\_matrix\_setcell3d  
    Matrix Module, 998  
jit\_matrix\_setinfo  
    Matrix Module, 999  
jit\_matrix\_setinfo\_ex  
    Matrix Module, 1000  
jit\_matrix\_setplane1d  
    Matrix Module, 1001  
jit\_matrix\_setplane2d  
    Matrix Module, 1002  
jit\_matrix\_setplane3d  
    Matrix Module, 1003  
jit\_mop\_free  
    MOP Module, 1038  
jit\_mop\_getinput  
    MOP Module, 1038  
jit\_mop\_getinputlist  
    MOP Module, 1039  
jit\_mop\_getoutput  
    MOP Module, 1039  
jit\_mop\_getoutputlist  
    MOP Module, 1040  
jit\_mop\_input\_nolink  
    MOP Module, 1040  
jit\_mop\_io\_free  
    MOP Module, 1041  
jit\_mop\_io\_getioproc  
    MOP Module, 1041  
jit\_mop\_io\_getmatrix  
    MOP Module, 1042  
jit\_mop\_io\_ioproc  
    MOP Module, 1042  
jit\_mop\_io\_matrix  
    MOP Module, 1043  
jit\_mop\_io\_new  
    MOP Module, 1044  
jit\_mop\_io\_newcopy  
    MOP Module, 1044  
jit\_mop\_io\_restrict\_dim  
    MOP Module, 1045  
jit\_mop\_io\_restrict\_planeCount  
    MOP Module, 1045  
jit\_mop\_io\_restrict\_type  
    MOP Module, 1046  
jit\_mop\_ioproc\_copy\_adapt  
    MOP Module, 1047  
jit\_mop\_ioproc\_copy\_trunc  
    MOP Module, 1048  
jit\_mop\_ioproc\_copy\_trunc\_zero  
    MOP Module, 1048  
jit\_mop\_ioproc\_tosym  
    MOP Module, 1049  
jit\_mop\_methodall  
    MOP Module, 1050  
jit\_mop\_new  
    MOP Module, 1050  
jit\_mop\_newcopy  
    MOP Module, 1051  
jit\_mop\_output\_nolink  
    MOP Module, 1052  
jit\_mop\_single\_planeCount  
    MOP Module, 1053  
jit\_mop\_single\_type  
    MOP Module, 1054  
jit\_newptr  
    Memory Module, 1035  
jit\_object\_attach  
    Object Module, 913  
jit\_object\_attr\_get  
    Object Module, 913  
jit\_object\_attr\_usercanget  
    Object Module, 914  
jit\_object\_attr\_usercanset  
    Object Module, 915  
jit\_object\_class  
    Object Module, 915

jit\_object\_classname  
     Object Module, [916](#)  
 jit\_object\_classname\_compare  
     Object Module, [917](#)  
 jit\_object\_detach  
     Object Module, [918](#)  
 jit\_object\_exportattrs  
     Object Module, [918](#)  
 jit\_object\_exportsummary  
     Object Module, [919](#)  
 jit\_object\_findregistered  
     Object Module, [920](#)  
 jit\_object\_findregisteredbyptr  
     Object Module, [921](#)  
 jit\_object\_free  
     Object Module, [921](#)  
 jit\_object\_getmethod  
     Object Module, [922](#)  
 jit\_object\_importattrs  
     Object Module, [923](#)  
 jit\_object\_method\_argsafe\_get  
     Object Module, [924](#)  
 jit\_object\_method\_imp  
     Object Module, [925](#)  
 jit\_object\_method\_typed  
     Object Module, [926](#)  
 jit\_object\_new\_imp  
     Object Module, [927](#)  
 jit\_object\_notify  
     Object Module, [927](#)  
 jit\_object\_register  
     Object Module, [928](#)  
 jit\_object\_unregister  
     Object Module, [929](#)  
 jit\_op\_vector\_abs\_float32  
     Operator Vector Module, [1090](#)  
 jit\_op\_vector\_abs\_float64  
     Operator Vector Module, [1090](#)  
 jit\_op\_vector\_abs\_long  
     Operator Vector Module, [1091](#)  
 jit\_op\_vector\_absdiff\_char  
     Operator Vector Module, [1091](#)  
 jit\_op\_vector\_absdiff\_float32  
     Operator Vector Module, [1092](#)  
 jit\_op\_vector\_absdiff\_float64  
     Operator Vector Module, [1092](#)  
 jit\_op\_vector\_absdiff\_long  
     Operator Vector Module, [1093](#)  
 jit\_op\_vector\_acos\_float32  
     Operator Vector Module, [1093](#)  
 jit\_op\_vector\_acos\_float64  
     Operator Vector Module, [1094](#)  
 jit\_op\_vector\_acosh\_float32  
     Operator Vector Module, [1094](#)  
 jit\_op\_vector\_acosh\_float64  
     Operator Vector Module, [1094](#)  
 jit\_op\_vector\_add\_char  
     Operator Vector Module, [1095](#)  
 jit\_op\_vector\_add\_float32  
     Operator Vector Module, [1095](#)  
 jit\_op\_vector\_add\_float64  
     Operator Vector Module, [1096](#)  
 jit\_op\_vector\_add\_long  
     Operator Vector Module, [1096](#)  
 jit\_op\_vector\_adds\_char  
     Operator Vector Module, [1097](#)  
 jit\_op\_vector\_and\_char  
     Operator Vector Module, [1097](#)  
 jit\_op\_vector\_and\_float32  
     Operator Vector Module, [1098](#)  
 jit\_op\_vector\_and\_float64  
     Operator Vector Module, [1098](#)  
 jit\_op\_vector\_and\_long  
     Operator Vector Module, [1099](#)  
 jit\_op\_vector\_asin\_float32  
     Operator Vector Module, [1099](#)  
 jit\_op\_vector\_asin\_float64  
     Operator Vector Module, [1099](#)  
 jit\_op\_vector\_asinh\_float32  
     Operator Vector Module, [1100](#)  
 jit\_op\_vector\_asinh\_float64  
     Operator Vector Module, [1100](#)  
 jit\_op\_vector\_atan2\_float32  
     Operator Vector Module, [1101](#)  
 jit\_op\_vector\_atan2\_float64  
     Operator Vector Module, [1101](#)  
 jit\_op\_vector\_atan\_float32  
     Operator Vector Module, [1102](#)  
 jit\_op\_vector\_atan\_float64  
     Operator Vector Module, [1102](#)  
 jit\_op\_vector\_atanh\_float32  
     Operator Vector Module, [1103](#)  
 jit\_op\_vector\_atanh\_float64  
     Operator Vector Module, [1103](#)  
 jit\_op\_vector\_avg\_char  
     Operator Vector Module, [1104](#)  
 jit\_op\_vector\_avg\_float32  
     Operator Vector Module, [1104](#)  
 jit\_op\_vector\_avg\_float64  
     Operator Vector Module, [1104](#)  
 jit\_op\_vector\_avg\_long  
     Operator Vector Module, [1105](#)  
 jit\_op\_vector\_bitand\_char  
     Operator Vector Module, [1105](#)  
 jit\_op\_vector\_bitand\_long  
     Operator Vector Module, [1106](#)  
 jit\_op\_vector\_bitnot\_char  
     Operator Vector Module, [1106](#)

jit\_op\_vector\_bitnot\_long  
    Operator Vector Module, 1107

jit\_op\_vector\_bitor\_char  
    Operator Vector Module, 1107

jit\_op\_vector\_bitor\_long  
    Operator Vector Module, 1108

jit\_op\_vector\_bitxor\_char  
    Operator Vector Module, 1108

jit\_op\_vector\_bitxor\_long  
    Operator Vector Module, 1109

jit\_op\_vector\_ceil\_float32  
    Operator Vector Module, 1109

jit\_op\_vector\_ceil\_float64  
    Operator Vector Module, 1109

jit\_op\_vector\_cos\_float32  
    Operator Vector Module, 1110

jit\_op\_vector\_cos\_float64  
    Operator Vector Module, 1110

jit\_op\_vector\_cosh\_float32  
    Operator Vector Module, 1111

jit\_op\_vector\_cosh\_float64  
    Operator Vector Module, 1111

jit\_op\_vector\_div\_char  
    Operator Vector Module, 1112

jit\_op\_vector\_div\_float32  
    Operator Vector Module, 1112

jit\_op\_vector\_div\_float64  
    Operator Vector Module, 1113

jit\_op\_vector\_div\_long  
    Operator Vector Module, 1113

jit\_op\_vector\_eq\_char  
    Operator Vector Module, 1114

jit\_op\_vector\_eq\_float32  
    Operator Vector Module, 1114

jit\_op\_vector\_eq\_float64  
    Operator Vector Module, 1115

jit\_op\_vector\_eq\_long  
    Operator Vector Module, 1115

jit\_op\_vector\_eqp\_char  
    Operator Vector Module, 1116

jit\_op\_vector\_eqp\_float32  
    Operator Vector Module, 1116

jit\_op\_vector\_eqp\_float64  
    Operator Vector Module, 1117

jit\_op\_vector\_eqp\_long  
    Operator Vector Module, 1117

jit\_op\_vector\_exp2\_float32  
    Operator Vector Module, 1117

jit\_op\_vector\_exp2\_float64  
    Operator Vector Module, 1118

jit\_op\_vector\_exp\_float32  
    Operator Vector Module, 1118

jit\_op\_vector\_exp\_float64  
    Operator Vector Module, 1119

jit\_op\_vector\_flipdiv\_char  
    Operator Vector Module, 1119

jit\_op\_vector\_flipdiv\_float32  
    Operator Vector Module, 1120

jit\_op\_vector\_flipdiv\_float64  
    Operator Vector Module, 1121

jit\_op\_vector\_flipdiv\_long  
    Operator Vector Module, 1121

jit\_op\_vector\_flipmod\_char  
    Operator Vector Module, 1122

jit\_op\_vector\_flipmod\_float32  
    Operator Vector Module, 1123

jit\_op\_vector\_flipmod\_float64  
    Operator Vector Module, 1124

jit\_op\_vector\_flipmod\_long  
    Operator Vector Module, 1124

jit\_op\_vector\_flippass\_char  
    Operator Vector Module, 1125

jit\_op\_vector\_flippass\_float32  
    Operator Vector Module, 1126

jit\_op\_vector\_flippass\_float64  
    Operator Vector Module, 1127

jit\_op\_vector\_flippass\_long  
    Operator Vector Module, 1127

jit\_op\_vector\_flipsub\_char  
    Operator Vector Module, 1128

jit\_op\_vector\_flipsub\_float32  
    Operator Vector Module, 1129

jit\_op\_vector\_flipsub\_long  
    Operator Vector Module, 1130

jit\_op\_vector\_floor\_float32  
    Operator Vector Module, 1130

jit\_op\_vector\_floor\_float64  
    Operator Vector Module, 1131

jit\_op\_vector\_fold\_float32  
    Operator Vector Module, 1131

jit\_op\_vector\_fold\_float64  
    Operator Vector Module, 1132

jit\_op\_vector\_gt\_char  
    Operator Vector Module, 1132

jit\_op\_vector\_gt\_float32  
    Operator Vector Module, 1133

jit\_op\_vector\_gt\_float64  
    Operator Vector Module, 1133

jit\_op\_vector\_gt\_long  
    Operator Vector Module, 1133

jit\_op\_vector\_gte\_char  
    Operator Vector Module, 1134

jit\_op\_vector\_gte\_float32  
    Operator Vector Module, 1134

jit\_op\_vector\_gte\_float64  
    Operator Vector Module, 1135

jit\_op\_vector\_gte\_long  
    Operator Vector Module, 1135

jit\_op\_vector\_gtep\_char  
     Operator Vector Module, 1136  
 jit\_op\_vector\_gtep\_float32  
     Operator Vector Module, 1136  
 jit\_op\_vector\_gtep\_float64  
     Operator Vector Module, 1137  
 jit\_op\_vector\_gtep\_long  
     Operator Vector Module, 1137  
 jit\_op\_vector\_gtp\_char  
     Operator Vector Module, 1138  
 jit\_op\_vector\_gtp\_float32  
     Operator Vector Module, 1138  
 jit\_op\_vector\_gtp\_float64  
     Operator Vector Module, 1138  
 jit\_op\_vector\_gtp\_long  
     Operator Vector Module, 1139  
 jit\_op\_vector\_hypot\_float32  
     Operator Vector Module, 1139  
 jit\_op\_vector\_hypot\_float64  
     Operator Vector Module, 1140  
 jit\_op\_vector\_log10\_float32  
     Operator Vector Module, 1140  
 jit\_op\_vector\_log10\_float64  
     Operator Vector Module, 1141  
 jit\_op\_vector\_log2\_float32  
     Operator Vector Module, 1141  
 jit\_op\_vector\_log2\_float64  
     Operator Vector Module, 1142  
 jit\_op\_vector\_log\_float32  
     Operator Vector Module, 1142  
 jit\_op\_vector\_log\_float64  
     Operator Vector Module, 1143  
 jit\_op\_vector\_lshift\_char  
     Operator Vector Module, 1143  
 jit\_op\_vector\_lshift\_long  
     Operator Vector Module, 1143  
 jit\_op\_vector\_lt\_char  
     Operator Vector Module, 1144  
 jit\_op\_vector\_lt\_float32  
     Operator Vector Module, 1144  
 jit\_op\_vector\_lt\_float64  
     Operator Vector Module, 1145  
 jit\_op\_vector\_lt\_long  
     Operator Vector Module, 1145  
 jit\_op\_vector\_lte\_char  
     Operator Vector Module, 1146  
 jit\_op\_vector\_lte\_float32  
     Operator Vector Module, 1146  
 jit\_op\_vector\_lte\_float64  
     Operator Vector Module, 1147  
 jit\_op\_vector\_lte\_long  
     Operator Vector Module, 1147  
 jit\_op\_vector\_ltep\_char  
     Operator Vector Module, 1148  
 jit\_op\_vector\_ltep\_float32  
     Operator Vector Module, 1148  
 jit\_op\_vector\_ltep\_float64  
     Operator Vector Module, 1148  
 jit\_op\_vector\_ltep\_long  
     Operator Vector Module, 1149  
 jit\_op\_vector\_ltp\_char  
     Operator Vector Module, 1149  
 jit\_op\_vector\_ltp\_float32  
     Operator Vector Module, 1150  
 jit\_op\_vector\_ltp\_float64  
     Operator Vector Module, 1150  
 jit\_op\_vector\_ltp\_long  
     Operator Vector Module, 1151  
 jit\_op\_vector\_max\_char  
     Operator Vector Module, 1151  
 jit\_op\_vector\_max\_float32  
     Operator Vector Module, 1152  
 jit\_op\_vector\_max\_float64  
     Operator Vector Module, 1152  
 jit\_op\_vector\_max\_long  
     Operator Vector Module, 1153  
 jit\_op\_vector\_min\_char  
     Operator Vector Module, 1153  
 jit\_op\_vector\_min\_float32  
     Operator Vector Module, 1153  
 jit\_op\_vector\_min\_float64  
     Operator Vector Module, 1154  
 jit\_op\_vector\_min\_long  
     Operator Vector Module, 1154  
 jit\_op\_vector\_mod\_char  
     Operator Vector Module, 1155  
 jit\_op\_vector\_mod\_float32  
     Operator Vector Module, 1155  
 jit\_op\_vector\_mod\_float64  
     Operator Vector Module, 1156  
 jit\_op\_vector\_mod\_long  
     Operator Vector Module, 1156  
 jit\_op\_vector\_mult\_char  
     Operator Vector Module, 1157  
 jit\_op\_vector\_mult\_float32  
     Operator Vector Module, 1157  
 jit\_op\_vector\_mult\_float64  
     Operator Vector Module, 1158  
 jit\_op\_vector\_mult\_long  
     Operator Vector Module, 1158  
 jit\_op\_vector\_neq\_char  
     Operator Vector Module, 1159  
 jit\_op\_vector\_neq\_float32  
     Operator Vector Module, 1159  
 jit\_op\_vector\_neq\_float64  
     Operator Vector Module, 1160  
 jit\_op\_vector\_neq\_long  
     Operator Vector Module, 1160

jit\_op\_vector\_neqp\_char  
    Operator Vector Module, 1160

jit\_op\_vector\_neqp\_float32  
    Operator Vector Module, 1161

jit\_op\_vector\_neqp\_float64  
    Operator Vector Module, 1161

jit\_op\_vector\_neqp\_long  
    Operator Vector Module, 1162

jit\_op\_vector\_not\_char  
    Operator Vector Module, 1162

jit\_op\_vector\_not\_float32  
    Operator Vector Module, 1163

jit\_op\_vector\_not\_float64  
    Operator Vector Module, 1163

jit\_op\_vector\_not\_long  
    Operator Vector Module, 1164

jit\_op\_vector\_or\_char  
    Operator Vector Module, 1164

jit\_op\_vector\_or\_float32  
    Operator Vector Module, 1165

jit\_op\_vector\_or\_float64  
    Operator Vector Module, 1165

jit\_op\_vector\_or\_long  
    Operator Vector Module, 1165

jit\_op\_vector\_pass\_char  
    Operator Vector Module, 1166

jit\_op\_vector\_pass\_float32  
    Operator Vector Module, 1166

jit\_op\_vector\_pass\_float64  
    Operator Vector Module, 1167

jit\_op\_vector\_pass\_long  
    Operator Vector Module, 1167

jit\_op\_vector\_pow\_float32  
    Operator Vector Module, 1168

jit\_op\_vector\_pow\_float64  
    Operator Vector Module, 1168

jit\_op\_vector\_round\_float32  
    Operator Vector Module, 1169

jit\_op\_vector\_round\_float64  
    Operator Vector Module, 1169

jit\_op\_vector\_rshift\_char  
    Operator Vector Module, 1170

jit\_op\_vector\_rshift\_long  
    Operator Vector Module, 1170

jit\_op\_vector\_sin\_float32  
    Operator Vector Module, 1171

jit\_op\_vector\_sin\_float64  
    Operator Vector Module, 1171

jit\_op\_vector\_sinh\_float32  
    Operator Vector Module, 1172

jit\_op\_vector\_sinh\_float64  
    Operator Vector Module, 1172

jit\_op\_vector\_sqrt\_float32  
    Operator Vector Module, 1172

jit\_op\_vector\_sqrt\_float64  
    Operator Vector Module, 1173

jit\_op\_vector\_sub\_char  
    Operator Vector Module, 1173

jit\_op\_vector\_sub\_float32  
    Operator Vector Module, 1174

jit\_op\_vector\_sub\_float64  
    Operator Vector Module, 1174

jit\_op\_vector\_sub\_long  
    Operator Vector Module, 1175

jit\_op\_vector\_subs\_char  
    Operator Vector Module, 1175

jit\_op\_vector\_tan\_float32  
    Operator Vector Module, 1176

jit\_op\_vector\_tan\_float64  
    Operator Vector Module, 1176

jit\_op\_vector\_tanh\_float32  
    Operator Vector Module, 1177

jit\_op\_vector\_tanh\_float64  
    Operator Vector Module, 1177

jit\_op\_vector\_trunc\_float32  
    Operator Vector Module, 1178

jit\_op\_vector\_trunc\_float64  
    Operator Vector Module, 1178

jit\_op\_vector\_wrap\_float32  
    Operator Vector Module, 1179

jit\_op\_vector\_wrap\_float64  
    Operator Vector Module, 1179

jit\_parallel\_ndim\_calc  
    Parallel Utility Module, 1055

jit\_parallel\_ndim\_simplecalc1  
    Parallel Utility Module, 1057

jit\_parallel\_ndim\_simplecalc2  
    Parallel Utility Module, 1058

jit\_parallel\_ndim\_simplecalc3  
    Parallel Utility Module, 1059

jit\_parallel\_ndim\_simplecalc4  
    Parallel Utility Module, 1060

jit\_post\_sym  
    Miscellaneous Utility Module, 934

jit\_rand  
    Miscellaneous Utility Module, 934

jit\_rand\_setseed  
    Miscellaneous Utility Module, 935

Jitter, 480

jkeyboard\_getcurrentmodifiers  
    Mouse and Keyboard, 544

jkeyboard\_getcurrentmodifiers\_realtime  
    Mouse and Keyboard, 545

jmonitor\_getdisplayrect  
    Monitors and Displays, 538

jmonitor\_getdisplayrect\_foralldisplays  
    Monitors and Displays, 538

jmonitor\_getdisplayrect\_forpoint

Monitors and Displays, 538  
**jmonitor\_getdisplayscalefactor**  
     Monitors and Displays, 539  
**jmonitor\_getdisplayscalefactor\_forpoint**  
     Monitors and Displays, 539  
**jmonitor\_getnumdisplays**  
     Monitors and Displays, 539  
**jmonitor\_scale\_pt**  
     Monitors and Displays, 540  
**jmonitor\_unscale\_pt**  
     Monitors and Displays, 540  
**JMOUSE\_CURSOR\_ARROW**  
     Mouse and Keyboard, 543  
**JMOUSE\_CURSOR\_COPYING**  
     Mouse and Keyboard, 543  
**JMOUSE\_CURSOR\_CROSSHAIR**  
     Mouse and Keyboard, 543  
**JMOUSE\_CURSOR\_DRAGGINGHAND**  
     Mouse and Keyboard, 544  
**JMOUSE\_CURSOR\_IBEAM**  
     Mouse and Keyboard, 543  
**JMOUSE\_CURSOR\_NONE**  
     Mouse and Keyboard, 543  
**JMOUSE\_CURSOR\_POINTINGHAND**  
     Mouse and Keyboard, 543  
**JMOUSE\_CURSOR\_RESIZE\_BOTTOMEDGE**  
     Mouse and Keyboard, 544  
**JMOUSE\_CURSOR\_RESIZE\_BOTTOMLEFTCORNER**  
     Mouse and Keyboard, 544  
**JMOUSE\_CURSOR\_RESIZE\_BOTTOMRIGHTCORNER**  
     Mouse and Keyboard, 544  
**JMOUSE\_CURSOR\_RESIZE\_FOURWAY**  
     Mouse and Keyboard, 544  
**JMOUSE\_CURSOR\_RESIZE\_LEFTEDGE**  
     Mouse and Keyboard, 544  
**JMOUSE\_CURSOR\_RESIZE\_LEFTRIGHT**  
     Mouse and Keyboard, 544  
**JMOUSE\_CURSOR\_RESIZE\_RIGHTEDGE**  
     Mouse and Keyboard, 544  
**JMOUSE\_CURSOR\_RESIZE\_TOPEDGE**  
     Mouse and Keyboard, 544  
**JMOUSE\_CURSOR\_RESIZE\_TOPLEFTCORNER**  
     Mouse and Keyboard, 544  
**JMOUSE\_CURSOR\_RESIZE\_TOPRIGHTCORNER**  
     Mouse and Keyboard, 544  
**JMOUSE\_CURSOR\_RESIZE\_UPDOWN**  
     Mouse and Keyboard, 544  
**JMOUSE\_CURSOR\_WAIT**  
     Mouse and Keyboard, 543  
**jmouse\_getposition\_global**  
     Mouse and Keyboard, 545  
**jmouse\_setcursor**  
     Mouse and Keyboard, 545  
**jmouse\_setposition\_box**

Mouse and Keyboard, 546  
**jmouse\_setposition\_global**  
     Mouse and Keyboard, 546  
**jmouse\_setposition\_view**  
     Mouse and Keyboard, 546  
**jpatcher**, 606  
     **jpatcher\_deleteobj**, 609  
     **jpatcher\_get\_bgcolor**, 609  
     **jpatcher\_get\_bghidden**, 609  
     **jpatcher\_get\_bglocked**, 610  
     **jpatcher\_get\_box**, 610  
     **jpatcher\_get\_count**, 611  
     **jpatcher\_get\_currentfileversion**, 611  
     **jpatcher\_get\_default\_fontface**, 611  
     **jpatcher\_get\_default\_fontname**, 612  
     **jpatcher\_get\_default\_fontsize**, 612  
     **jpatcher\_get\_defrect**, 612  
     **jpatcher\_get\_dirty**, 613  
     **jpatcher\_get\_editing\_bgcolor**, 613  
     **jpatcher\_get\_fghidden**, 614  
     **jpatcher\_get\_filename**, 614  
     **jpatcher\_get\_filepath**, 614  
     **jpatcher\_get\_fileversion**, 615  
     **jpatcher\_get\_firstline**, 615  
     **jpatcher\_get\_firstobject**, 616  
     **jpatcher\_get\_firstview**, 616  
     **jpatcher\_get\_gridsize**, 617  
     **jpatcher\_get\_hubholder**, 617  
     **jpatcher\_get\_lastobject**, 618  
     **jpatcher\_get\_locked\_bgcolor**, 618  
     **jpatcher\_get\_name**, 619  
     **jpatcher\_get\_parentpatcher**, 619  
     **jpatcher\_get\_presentation**, 619  
     **jpatcher\_get\_rect**, 620  
     **jpatcher\_get\_title**, 620  
     **jpatcher\_get\_toppatcher**, 620  
     **jpatcher\_is\_patch**, 621  
     **jpatcher\_set\_bgcolor**, 621  
     **jpatcher\_set\_bghidden**, 622  
     **jpatcher\_set\_bglocked**, 622  
     **jpatcher\_set\_defrect**, 622  
     **jpatcher\_set\_dirty**, 623  
     **jpatcher\_set\_editing\_bgcolor**, 623  
     **jpatcher\_set\_fghidden**, 624  
     **jpatcher\_set\_gridsize**, 624  
     **jpatcher\_set\_locked**, 625  
     **jpatcher\_set\_locked\_bgcolor**, 625  
     **jpatcher\_set\_presentation**, 625  
     **jpatcher\_set\_rect**, 626  
     **jpatcher\_set\_title**, 626  
     **jpatcher\_uniqueboxname**, 627  
**jpatcher\_deleteobj**  
     **jpatcher**, 609  
**jpatcher\_get\_bgcolor**

jpacher, 609  
jpacher\_get\_bghidden  
    jpacher, 609  
jpacher\_get\_bglocked  
    jpacher, 610  
jpacher\_get\_box  
    jpacher, 610  
jpacher\_get\_count  
    jpacher, 611  
jpacher\_get\_currentfileversion  
    jpacher, 611  
jpacher\_get\_default\_fontface  
    jpacher, 611  
jpacher\_get\_default\_fontname  
    jpacher, 612  
jpacher\_get\_default\_fontsize  
    jpacher, 612  
jpacher\_get\_defrect  
    jpacher, 612  
jpacher\_get\_dirty  
    jpacher, 613  
jpacher\_get\_editing\_bgcolor  
    jpacher, 613  
jpacher\_get\_fghidden  
    jpacher, 614  
jpacher\_get\_filename  
    jpacher, 614  
jpacher\_get\_filepath  
    jpacher, 614  
jpacher\_get\_fileversion  
    jpacher, 615  
jpacher\_get\_firstline  
    jpacher, 615  
jpacher\_get\_firstobject  
    jpacher, 616  
jpacher\_get\_firstview  
    jpacher, 616  
jpacher\_get\_grayscale  
    jpacher, 617  
jpacher\_get\_hubholder  
    jpacher, 617  
jpacher\_get\_lastobject  
    jpacher, 618  
jpacher\_get\_locked\_bgcolor  
    jpacher, 618  
jpacher\_get\_name  
    jpacher, 619  
jpacher\_get\_parentpatcher  
    jpacher, 619  
jpacher\_get\_presentation  
    jpacher, 619  
jpacher\_get\_rect  
    jpacher, 620  
jpacher\_get\_title

jpacher, 620  
jpacher\_get\_toppatcher  
    jpacher, 620  
jpacher\_is\_patch  
    jpacher, 621  
jpacher\_set\_bgcolor  
    jpacher, 621  
jpacher\_set\_bghidden  
    jpacher, 622  
jpacher\_set\_bglocked  
    jpacher, 622  
jpacher\_set\_defrect  
    jpacher, 622  
jpacher\_set\_dirty  
    jpacher, 623  
jpacher\_set\_editing\_bgcolor  
    jpacher, 623  
jpacher\_set\_fghidden  
    jpacher, 624  
jpacher\_set\_grayscale  
    jpacher, 624  
jpacher\_set\_locked  
    jpacher, 625  
jpacher\_set\_locked\_bgcolor  
    jpacher, 625  
jpacher\_set\_presentation  
    jpacher, 625  
jpacher\_set\_rect  
    jpacher, 626  
jpacher\_set\_title  
    jpacher, 626  
jpacher\_uniqueboxname  
    jpacher, 627  
jpacherview, 665  
    patcherview\_canvas\_to\_screen, 667  
    patcherview\_findpatcherview, 667  
    patcherview\_get\_jgraphics, 668  
    patcherview\_get\_locked, 668  
    patcherview\_get\_nextview, 668  
    patcherview\_get\_patch, 670  
    patcherview\_get\_presentation, 670  
    patcherview\_get\_rect, 670  
    patcherview\_get\_topview, 671  
    patcherview\_get\_visible, 671  
    patcherview\_get\_zoomfactor, 672  
    patcherview\_screen\_to\_canvas, 672  
    patcherview\_set\_locked, 673  
    patcherview\_set\_presentation, 673  
    patcherview\_set\_rect, 673  
    patcherview\_set\_visible, 674  
    patcherview\_set\_zoomfactor, 674  
jpatchline, 660  
    jpatchline\_get\_box1, 661  
    jpatchline\_get\_box2, 661

jpatchline\_get\_color, 661  
 jpatchline\_get\_endpoint, 662  
 jpatchline\_get\_hidden, 662  
 jpatchline\_get\_inletnum, 663  
 jpatchline\_get\_nextline, 663  
 jpatchline\_get\_nummidpoints, 663  
 jpatchline\_get\_outletnum, 664  
 jpatchline\_get\_startpoint, 664  
 jpatchline\_set\_color, 665  
 jpatchline\_set\_hidden, 665  
 jpatchline\_get\_box1  
     jpatchline, 661  
 jpatchline\_get\_box2  
     jpatchline, 661  
 jpatchline\_get\_color  
     jpatchline, 661  
 jpatchline\_get\_endpoint  
     jpatchline, 662  
 jpatchline\_get\_hidden  
     jpatchline, 662  
 jpatchline\_get\_inletnum  
     jpatchline, 663  
 jpatchline\_get\_nextline  
     jpatchline, 663  
 jpatchline\_get\_nummidpoints  
     jpatchline, 663  
 jpatchline\_get\_outletnum  
     jpatchline, 664  
 jpatchline\_get\_startpoint  
     jpatchline, 664  
 jpatchline\_set\_color  
     jpatchline, 665  
 jpatchline\_set\_hidden  
     jpatchline, 665  
 JPattern, 800  
 jpopupmenu\_additem  
     Popup Menus, 836  
 jpopupmenu\_addseparator  
     Popup Menus, 836  
 jpopupmenu\_addsubmenu  
     Popup Menus, 837  
 jpopupmenu\_clear  
     Popup Menus, 837  
 jpopupmenu\_create  
     Popup Menus, 837  
 jpopupmenu\_destroy  
     Popup Menus, 838  
 jpopupmenu\_popup  
     Popup Menus, 838  
 jpopupmenu\_popup\_abovebox  
     Popup Menus, 838  
 jpopupmenu\_popup\_aboverect  
     Popup Menus, 839  
 jpopupmenu\_popup\_belowrect  
  
 Popup Menus, 839  
 jpopupmenu\_popup\_nearbox  
     Popup Menus, 840  
 jpopupmenu\_popup\_nearbox\_with\_options  
     Popup Menus, 840  
 jpopupmenu\_setcolors  
     Popup Menus, 841  
 jpopupmenuSetFont  
     Popup Menus, 841  
 jrgba\_attr\_get  
     Colors, 803  
 jrgba\_attr\_set  
     Colors, 803  
 jrgba\_compare  
     Colors, 804  
 jrgba\_copy  
     Colors, 804  
 jrgba\_set  
     Colors, 804  
 jrgba\_to\_atoms  
     Colors, 805  
 JSurface, 766  
     jgraphics\_create, 768  
     jgraphics\_get\_resource\_data, 768  
     jgraphics\_image\_surface\_clear, 769  
     jgraphics\_image\_surface\_create, 769  
     jgraphics\_image\_surface\_create\_for\_data, 770  
     jgraphics\_image\_surface\_create\_from\_file, 770  
     jgraphics\_image\_surface\_create\_from\_filedata, 771  
     jgraphics\_image\_surface\_create\_from\_resource, 771  
     jgraphics\_image\_surface\_create\_referenced, 772  
     jgraphics\_image\_surface\_draw, 772  
     jgraphics\_image\_surface\_draw\_fast, 773  
     jgraphics\_image\_surface\_get\_height, 773  
     jgraphics\_image\_surface\_get\_pixel, 774  
     jgraphics\_image\_surface\_get\_width, 774  
     jgraphics\_image\_surface\_scroll, 775  
     jgraphics\_image\_surface\_set\_pixel, 775  
     jgraphics\_image\_surface\_writejpeg, 776  
     jgraphics\_image\_surface\_writepng, 776  
     jgraphics\_surface\_destroy, 776  
     jgraphics\_surface\_reference, 777  
     jgraphics\_write\_image\_surface\_to\_filedata, 777  
 jsvg\_create\_from\_file  
     Scalable Vector Graphics, 778  
 jsvg\_create\_from\_resource  
     Scalable Vector Graphics, 779  
 jsvg\_create\_from\_xmlstring  
     Scalable Vector Graphics, 779  
 jsvg\_destroy  
     Scalable Vector Graphics, 780  
 jsvg\_get\_size  
     Scalable Vector Graphics, 780

jsvg\_render  
Scalable Vector Graphics, 780

jtextlayout\_create  
TextLayout, 830

jtextlayout\_createpath  
TextLayout, 830

jtextlayout\_destroy  
TextLayout, 830

jtextlayout\_draw  
TextLayout, 831

jtextlayout\_getchar  
TextLayout, 831

jtextlayout\_getcharbox  
TextLayout, 831

jtextlayout\_getnumchars  
TextLayout, 832

jtextlayout\_measuretext  
TextLayout, 832

jtextlayout\_set  
TextLayout, 833

jtextlayout\_settext  
TextLayout, 833

jtextlayout\_settextcolor  
TextLayout, 834

jtextlayout\_withbgcolor  
TextLayout, 834

jwind\_getactive  
Windows, 541

jwind\_getat  
Windows, 541

jwind\_getcount  
Windows, 542

Linked List, 362

linklist\_append, 365

linklist\_chuck, 366

linklist\_chuckindex, 366

linklist\_chuckobject, 367

linklist\_clear, 367

linklist\_deleteindex, 368

linklist\_deleteobject, 368

linklist\_findall, 369

linklist\_findfirst, 370

linklist\_flags, 371

linklist\_funall, 371

linklist\_funall\_break, 372

linklist\_funindex, 372

linklist\_getflags, 373

linklist\_getindex, 373

linklist\_getsize, 374

linklist\_insert\_sorted, 374

linklist\_insertafterobjptr, 375

linklist\_insertbeforeobjptr, 375

linklist\_insertindex, 375

linklist\_last, 376

linklist\_makearray, 376

linklist\_match, 377

linklist\_methodall, 377

linklist\_methodindex, 378

linklist\_moveafterobjptr, 379

linklist\_movebeforeobjptr, 379

linklist\_new, 379

linklist\_next, 380

linklist\_objptr2index, 380

linklist\_prev, 382

linklist\_READONLY, 382

linklist\_reverse, 383

linklist\_rotate, 383

linklist\_shuffle, 383

linklist\_sort, 384

linklist\_substitute, 384

linklist\_swap, 385

Linked List Module, 936

jit\_linklist\_append, 937

jit\_linklist\_chuck, 938

jit\_linklist\_chuckindex, 939

jit\_linklist\_clear, 940

jit\_linklist\_deleteindex, 941

jit\_linklist\_findall, 942

jit\_linklist\_findcount, 943

jit\_linklist\_findfirst, 944

jit\_linklist\_getindex, 945

jit\_linklist\_getsize, 946

jit\_linklist\_insertindex, 947

jit\_linklist\_makearray, 947

jit\_linklist\_methodall, 948

jit\_linklist\_methodindex, 949

jit\_linklist\_new, 950

jit\_linklist\_objptr2index, 950

jit\_linklist\_reverse, 951

jit\_linklist\_rotate, 952

jit\_linklist\_shuffle, 953

jit\_linklist\_sort, 953

jit\_linklist\_swap, 954

linklist\_append  
Linked List, 365

linklist\_chuck  
Linked List, 366

linklist\_chuckindex  
Linked List, 366

linklist\_chuckobject  
Linked List, 367

linklist\_clear  
Linked List, 367

linklist\_deleteindex  
Linked List, 368

linklist\_deleteobject  
Linked List, 368

**linklist\_findall**  
 Linked List, 369  
**linklist\_findfirst**  
 Linked List, 370  
**linklist\_flags**  
 Linked List, 371  
**linklist\_funall**  
 Linked List, 371  
**linklist\_funall\_break**  
 Linked List, 372  
**linklist\_funindex**  
 Linked List, 372  
**linklist\_getflags**  
 Linked List, 373  
**linklist\_getindex**  
 Linked List, 373  
**linklist\_getsize**  
 Linked List, 374  
**linklist\_insert\_sorted**  
 Linked List, 374  
**linklist\_insertafterobjptr**  
 Linked List, 375  
**linklist\_insertbeforeobjptr**  
 Linked List, 375  
**linklist\_insertindex**  
 Linked List, 375  
**linklist\_last**  
 Linked List, 376  
**linklist\_makearray**  
 Linked List, 376  
**linklist\_match**  
 Linked List, 377  
**linklist\_methodall**  
 Linked List, 377  
**linklist\_methodindex**  
 Linked List, 378  
**linklist\_moveafterobjptr**  
 Linked List, 379  
**linklist\_movebeforeobjptr**  
 Linked List, 379  
**linklist\_new**  
 Linked List, 379  
**linklist\_next**  
 Linked List, 380  
**linklist\_objptr2index**  
 Linked List, 380  
**linklist\_prev**  
 Linked List, 382  
**linklist\_READONLY**  
 Linked List, 382  
**linklist\_reverse**  
 Linked List, 383  
**linklist\_rotate**  
 Linked List, 383

**linklist\_shuffle**  
 Linked List, 383  
**linklist\_sort**  
 Linked List, 384  
**linklist\_substitute**  
 Linked List, 384  
**linklist\_swap**  
 Linked List, 385  
**listout**  
 Inlets and Outlets, 272  
**Loading Max Files**, 534  
 fileload, 535  
 intload, 535  
 readtohandle, 536  
 stringload, 536  
**locatefile**  
 Files and Folders, 456  
**locatefile\_extended**  
 Files and Folders, 456  
**locatefiletype**  
 Files and Folders, 457

**Math Module**, 955  
 jit\_math\_acos, 957  
 jit\_math\_acosh, 958  
 jit\_math\_asin, 958  
 jit\_math\_asinh, 958  
 jit\_math\_atan, 959  
 jit\_math\_atan2, 959  
 jit\_math\_atanh, 960  
 jit\_math\_ceil, 960  
 jit\_math\_cos, 960  
 jit\_math\_cosh, 961  
 jit\_math\_exp, 961  
 jit\_math\_exp2, 961  
 jit\_math\_expm1, 962  
 jit\_math\_fast\_acos, 962  
 jit\_math\_fast\_asin, 963  
 jit\_math\_fast\_atan, 964  
 jit\_math\_fast\_cos, 964  
 jit\_math\_fast\_invsqrt, 965  
 jit\_math\_fast\_sin, 965  
 jit\_math\_fast\_sqrt, 965  
 jit\_math\_fast\_tan, 967  
 jit\_math\_floor, 967  
 jit\_math\_fmod, 967  
 jit\_math\_fold, 968  
 jit\_math\_hypot, 968  
 jit\_math\_is\_finite, 969  
 jit\_math\_is\_nan, 969  
 jit\_math\_is\_poweroftwo, 970  
 jit\_math\_is\_valid, 970  
 jit\_math\_j1, 971  
 jit\_math\_j1\_0, 972

jit\_math\_log, 972  
jit\_math\_log10, 973  
jit\_math\_log2, 973  
jit\_math\_p1, 974  
jit\_math\_pow, 974  
jit\_math\_q1, 974  
jit\_math\_round, 975  
jit\_math\_roundup\_poweroftwo, 975  
jit\_math\_sin, 976  
jit\_math\_sinh, 976  
jit\_math\_sqrt, 976  
jit\_math\_tan, 977  
jit\_math\_tanh, 977  
jit\_math\_trunc, 978  
jit\_math\_wrap, 978  
Matrix Module, 979  
  jit\_linklist\_free, 980  
  jit\_matrix\_clear, 981  
  jit\_matrix\_data, 981  
  jit\_matrix\_exprfill, 982  
  jit\_matrix\_fillplane, 983  
  jit\_matrix\_free, 984  
  jit\_matrix\_freedata, 985  
  jit\_matrix\_frommatrix, 986  
  jit\_matrix\_getcell, 986  
  jit\_matrix\_getdata, 988  
  jit\_matrix\_getinfo, 988  
  jit\_matrix\_info\_default, 989  
  jit\_matrix\_jit\_gl\_texture, 990  
  jit\_matrix\_new, 990  
  jit\_matrix\_newcopy, 991  
  jit\_matrix\_op, 992  
  jit\_matrix\_setall, 993  
  jit\_matrix\_setcell, 995  
  jit\_matrix\_setcell1d, 996  
  jit\_matrix\_setcell2d, 997  
  jit\_matrix\_setcell3d, 998  
  jit\_matrix\_setinfo, 999  
  jit\_matrix\_setinfo\_ex, 1000  
  jit\_matrix\_setplane1d, 1001  
  jit\_matrix\_setplane2d, 1002  
  jit\_matrix\_setplane3d, 1003  
MAX  
  Miscellaneous, 505  
Max Wrapper Module, 1004  
  max\_addmethod\_defer, 1006  
  max\_addmethod\_defer\_low, 1007  
  max\_addmethod\_usurp, 1007  
  max\_addmethod\_usurp\_low, 1008  
  max\_jit\_attr\_args, 1008  
  max\_jit\_attr\_args\_offset, 1009  
  max\_jit\_attr\_get, 1010  
  max\_jit\_attr\_getdump, 1011  
  max\_jit\_attr\_set, 1012  
  max\_jit\_classex\_addattr, 1013  
  max\_jit\_classex\_setup, 1014  
  max\_jit\_classex\_standard\_wrap, 1015  
  max\_jit\_obex\_adornment\_get, 1016  
  max\_jit\_obex\_attr\_get, 1017  
  max\_jit\_obex\_attr\_set, 1018  
  max\_jit\_obex\_dumpout, 1019  
  max\_jit\_obex\_dumpout\_get, 1020  
  max\_jit\_obex\_dumpout\_set, 1020  
  max\_jit\_obex\_free, 1021  
  max\_jit\_obex\_gimmeback, 1021  
  max\_jit\_obex\_gimmeback\_dumpout, 1022  
  max\_jit\_obex\_inletnumber\_get, 1023  
  max\_jit\_obex\_inletnumber\_set, 1024  
  max\_jit\_obex\_jitob\_get, 1024  
  max\_jit\_obex\_jitob\_set, 1025  
  max\_jit\_obex\_new, 1025  
  max\_jit\_obex\_proxy\_new, 1026  
max\_addmethod\_defer  
  Max Wrapper Module, 1006  
max\_addmethod\_defer\_low  
  Max Wrapper Module, 1007  
max\_addmethod\_usurp  
  Max Wrapper Module, 1007  
max\_addmethod\_usurp\_low  
  Max Wrapper Module, 1008  
MAX\_ERR\_DUPLICATE  
  Miscellaneous, 507  
MAX\_ERR\_GENERIC  
  Miscellaneous, 507  
MAX\_ERR\_INVALID\_PTR  
  Miscellaneous, 507  
MAX\_ERR\_NONE  
  Miscellaneous, 507  
MAX\_ERR\_OUT\_OF\_MEM  
  Miscellaneous, 507  
max\_jit\_attr\_args  
  Max Wrapper Module, 1008  
max\_jit\_attr\_args\_offset  
  Max Wrapper Module, 1009  
max\_jit\_attr\_get  
  Max Wrapper Module, 1010  
max\_jit\_attr\_getdump  
  Max Wrapper Module, 1011  
max\_jit\_attr\_set  
  Max Wrapper Module, 1012  
max\_jit\_classex\_addattr  
  Max Wrapper Module, 1013  
max\_jit\_classex\_mop\_mproc  
  MOP Max Wrapper Module, 1063  
max\_jit\_classex\_mop\_wrap  
  MOP Max Wrapper Module, 1063  
max\_jit\_classex\_setup  
  Max Wrapper Module, 1014

max\_jit\_classex\_standard\_wrap  
     Max Wrapper Module, 1015

max\_jit\_mop\_adapt\_matrix\_all  
     MOP Max Wrapper Module, 1065

max\_jit\_mop\_assist  
     MOP Max Wrapper Module, 1066

max\_jit\_mop\_bang  
     MOP Max Wrapper Module, 1067

max\_jit\_mop\_clear  
     MOP Max Wrapper Module, 1068

max\_jit\_mop\_free  
     MOP Max Wrapper Module, 1068

max\_jit\_mop\_get\_io\_by\_name  
     MOP Max Wrapper Module, 1069

max\_jit\_mop\_getinput  
     MOP Max Wrapper Module, 1070

max\_jit\_mop\_getoutput  
     MOP Max Wrapper Module, 1070

max\_jit\_mop\_getoutputmode  
     MOP Max Wrapper Module, 1071

max\_jit\_mop\_inputs  
     MOP Max Wrapper Module, 1072

max\_jit\_mop\_jit\_matrix  
     MOP Max Wrapper Module, 1073

max\_jit\_mop\_matrix\_args  
     MOP Max Wrapper Module, 1074

max\_jit\_mop\_matrixout\_new  
     MOP Max Wrapper Module, 1075

max\_jit\_mop\_notify  
     MOP Max Wrapper Module, 1075

max\_jit\_mop\_outputmatrix  
     MOP Max Wrapper Module, 1076

max\_jit\_mop\_outputs  
     MOP Max Wrapper Module, 1077

max\_jit\_mop\_setup  
     MOP Max Wrapper Module, 1078

max\_jit\_mop\_setup\_simple  
     MOP Max Wrapper Module, 1079

max\_jit\_mop\_variable\_addinputs  
     MOP Max Wrapper Module, 1080

max\_jit\_mop\_variable\_addoutputs  
     MOP Max Wrapper Module, 1081

max\_jit\_obex\_adornment\_get  
     Max Wrapper Module, 1016

max\_jit\_obex\_attr\_get  
     Max Wrapper Module, 1017

max\_jit\_obex\_attr\_set  
     Max Wrapper Module, 1018

max\_jit\_obex\_dumpout  
     Max Wrapper Module, 1019

max\_jit\_obex\_dumpout\_get  
     Max Wrapper Module, 1020

max\_jit\_obex\_dumpout\_set  
     Max Wrapper Module, 1020

max\_jit\_obex\_free  
     Max Wrapper Module, 1021

max\_jit\_obex\_gimmeback  
     Max Wrapper Module, 1021

max\_jit\_obex\_gimmeback\_dumpout  
     Max Wrapper Module, 1022

max\_jit\_obex\_inletnumber\_get  
     Max Wrapper Module, 1023

max\_jit\_obex\_inletnumber\_set  
     Max Wrapper Module, 1024

max\_jit\_obex\_jitob\_get  
     Max Wrapper Module, 1024

max\_jit\_obex\_jitob\_set  
     Max Wrapper Module, 1025

max\_jit\_obex\_new  
     Max Wrapper Module, 1025

max\_jit\_obex\_proxy\_new  
     Max Wrapper Module, 1026

maxversion  
     Miscellaneous, 511

Memory Management, 487

- disposhandle, 489
- freebytes, 490
- freebytes16, 490
- getbytes, 491
- getbytes16, 491
- growhandle, 492
- MM\_UNIFIED, 489
- newhandle, 492
- sysmem\_copyptr, 493
- sysmem\_freehandle, 493
- sysmem\_freeptr, 493
- sysmem\_handlesize, 494
- sysmem\_lockhandle, 494
- sysmem\_newhandle, 495
- sysmem\_newhandleclear, 495
- sysmem\_newptr, 496
- sysmem\_newptrclear, 496
- sysmem\_nullterminatehandle, 497
- sysmem\_ptrandhand, 497
- sysmem\_ptrbeforehand, 497
- sysmem\_ptrsize, 498
- sysmem\_resizehandle, 498
- sysmem\_resizeptr, 499
- sysmem\_resizeptrclear, 499

Memory Module, 1027

- jit\_copy\_bytes, 1028
- jit\_disposeptr, 1029
- jit\_freebytes, 1029
- jit\_freetem, 1030
- jit\_getbytes, 1030
- jit\_handle\_free, 1031
- jit\_handle\_lock, 1032
- jit\_handle\_new, 1033

jit\_handle\_size\_get, 1034  
jit\_handle\_size\_set, 1034  
jit\_newptr, 1035

MIN  
  Miscellaneous, 506

Miscellaneous, 500  
  aaCancel, 507  
  aaNo, 507  
  aaYes, 507  
  BEGIN\_USING\_C\_LINKAGE, 504  
  calcoffset, 504  
  CLAMP, 504  
  e\_max\_errorcodes, 507  
  e\_max\_wind\_advise\_result, 507  
  error\_subscribe, 508  
  error\_sym, 508  
  error\_unsubscribe, 508  
  globalsymbol\_bind, 509  
  globalsymbol\_dereference, 509  
  globalsymbol\_reference, 510  
  globalsymbol\_unbind, 510  
  INRANGE, 505  
  MAX, 505  
  MAX\_ERR\_DUPLICATE, 507  
  MAX\_ERR\_GENERIC, 507  
  MAX\_ERR\_INVALID\_PTR, 507  
  MAX\_ERR\_NONE, 507  
  MAX\_ERR\_OUT\_OF\_MEM, 507  
  maxversion, 511  
  MIN, 506  
  object\_obex\_quickref, 511  
  post\_sym, 511  
  quittask\_install, 512  
  quittask\_remove, 512  
  snprintf\_zero, 512  
  strncat\_zero, 513  
  strncpy\_zero, 513  
  structmembersize, 506  
  symbol\_stripquotes, 514  
  symbol\_unique, 514  
  symbolarray\_sort, 514  
  wind\_advise, 515  
  wind\_setcursor, 515

Miscellaneous Utility Module, 930  
  jit\_err\_from\_max\_err, 931  
  jit\_error\_code, 931  
  jit\_error\_sym, 932  
  jit\_global\_critical\_enter, 933  
  jit\_global\_critical\_exit, 933  
  jit\_post\_sym, 934  
  jit\_rand, 934  
  jit\_rand\_setseed, 935  
  swapf32, 935  
  swapf64, 936

MM\_UNIFIED  
  Memory Management, 489

Monitors and Displays, 537  
  jmonitor\_getdisplayrect, 538  
  jmonitor\_getdisplayrect\_foralldisplays, 538  
  jmonitor\_getdisplayrect\_forpoint, 538  
  jmonitor\_getdisplayscalefactor, 539  
  jmonitor\_getdisplayscalefactor\_forpoint, 539  
  jmonitor\_getnumdisplays, 539  
  jmonitor\_scale\_pt, 540  
  jmonitor\_unscale\_pt, 540

MOP Max Wrapper Module, 1061  
  max\_jit\_classex\_mop\_mproc, 1063  
  max\_jit\_classex\_mop\_wrap, 1063  
  max\_jit\_mop\_adapt\_matrix\_all, 1065  
  max\_jit\_mop\_assist, 1066  
  max\_jit\_mop\_bang, 1067  
  max\_jit\_mop\_clear, 1068  
  max\_jit\_mop\_free, 1068  
  max\_jit\_mop\_get\_io\_by\_name, 1069  
  max\_jit\_mop\_getinput, 1070  
  max\_jit\_mop\_getoutput, 1070  
  max\_jit\_mop\_getoutputmode, 1071  
  max\_jit\_mop\_inputs, 1072  
  max\_jit\_mop\_jit\_matrix, 1073  
  max\_jit\_mop\_matrix\_args, 1074  
  max\_jit\_mop\_matrixout\_new, 1075  
  max\_jit\_mop\_notify, 1075  
  max\_jit\_mop\_outputmatrix, 1076  
  max\_jit\_mop\_outputs, 1077  
  max\_jit\_mop\_setup, 1078  
  max\_jit\_mop\_setup\_simple, 1079  
  max\_jit\_mop\_variable\_addinputs, 1080  
  max\_jit\_mop\_variable\_addoutputs, 1081

MOP Module, 1036  
  jit\_mop\_free, 1038  
  jit\_mop\_getinput, 1038  
  jit\_mop\_getinputlist, 1039  
  jit\_mop\_getoutput, 1039  
  jit\_mop\_getoutputlist, 1040  
  jit\_mop\_input\_nolink, 1040  
  jit\_mop\_io\_free, 1041  
  jit\_mop\_io\_getioproc, 1041  
  jit\_mop\_io\_getmatrix, 1042  
  jit\_mop\_io\_ioproc, 1042  
  jit\_mop\_io\_matrix, 1043  
  jit\_mop\_io\_new, 1044  
  jit\_mop\_io\_newcopy, 1044  
  jit\_mop\_io\_restrict\_dim, 1045  
  jit\_mop\_io\_restrict\_planeCount, 1045  
  jit\_mop\_io\_restrict\_type, 1046  
  jit\_mop\_ioproc\_copy\_adapt, 1047  
  jit\_mop\_ioproc\_copy\_trunc, 1048  
  jit\_mop\_ioproc\_copy\_trunc\_zero, 1048

jit\_mop\_ioproc\_tosym, 1049  
 jit\_mop\_methodall, 1050  
 jit\_mop\_new, 1050  
 jit\_mop\_newcopy, 1051  
 jit\_mop\_output\_nolink, 1052  
 jit\_mop\_single\_planecount, 1053  
 jit\_mop\_single\_type, 1054  
**Mouse and Keyboard**, 542  
 eAltKey, 544  
 eAutoRepeat, 544  
 eCapsLock, 544  
 eCommandKey, 544  
 eControlKey, 544  
 eLeftButton, 544  
 eMiddleButton, 544  
 ePopupMenu, 544  
 eRightButton, 544  
 eShiftKey, 544  
 jkeyboard\_getcurrentmodifiers, 544  
 jkeyboard\_getcurrentmodifiers\_realtime, 545  
 JMOUSE\_CURSOR\_ARROW, 543  
 JMOUSE\_CURSOR\_COPYING, 543  
 JMOUSE\_CURSOR\_CROSSLHAIR, 543  
 JMOUSE\_CURSOR\_DRAGGINGHAND, 544  
 JMOUSE\_CURSOR\_IBeam, 543  
 JMOUSE\_CURSOR\_NONE, 543  
 JMOUSE\_CURSOR\_POINTINGHAND, 543  
 JMOUSE\_CURSOR\_RESIZE\_BOTTOMEDGE, 544  
 JMOUSE\_CURSOR\_RESIZE\_BOTTOMLEFTCORNER, 544  
 JMOUSE\_CURSOR\_RESIZE\_BOTTOMRIGHTCORNER, 544  
 JMOUSE\_CURSOR\_RESIZE\_FOURWAY, 544  
 JMOUSE\_CURSOR\_RESIZE\_LEFTEDGE, 544  
 JMOUSE\_CURSOR\_RESIZE\_LEFTRIGHT, 544  
 JMOUSE\_CURSOR\_RESIZE\_RIGHTEDGE, 544  
 JMOUSE\_CURSOR\_RESIZE\_TOPEDGE, 544  
 JMOUSE\_CURSOR\_RESIZE\_TOPLEFTCORNER, 544  
 JMOUSE\_CURSOR\_RESIZE\_TOPRIGHTCORNER, 544  
 JMOUSE\_CURSOR\_RESIZE\_UPDOWN, 544  
 JMOUSE\_CURSOR\_WAIT, 543  
 jmouse\_getposition\_global, 545  
 jmouse\_setcursor, 545  
 jmouse\_setposition\_box, 546  
 jmouse\_setposition\_global, 546  
 jmouse\_setposition\_view, 546  
 t\_jmouse\_cursortype, 543  
 t\_modifiers, 544  
**MSP**, 547  
 class\_dspinit, 551  
 class\_dspinitjbox, 552  
 dsp\_add, 552  
 dsp\_advv, 553  
 PI, 549  
 PIOVERTWO, 549  
 sys\_getblksize, 553  
 sys\_getdspobjdspstate, 553  
 sys\_getdspstate, 554  
 sys\_getmaxblksize, 554  
 sys\_getsr, 554  
 SYS\_MAXBLKSIZE, 551  
 t\_double, 550  
 t\_float, 550  
 t\_perfroutine, 556  
 t\_sample, 550  
 t\_vptr, 550  
 TWOPI, 550  
 vptr, 551  
 z\_dsp\_free, 555  
 z\_dsp\_setup, 555  
**Mutexes**, 732  
 systhread\_mutex\_free, 733  
 systhread\_mutex\_lock, 733  
 systhread\_mutex\_new, 733  
 systhread\_mutex\_newlock, 734  
 systhread\_mutex\_trylock, 734  
 systhread\_mutex\_unlock, 735  
 newhandle  
     Memory Management, 492  
 newinstance  
     Old-Style Classes, 264  
 newobject  
     Old-Style Classes, 265  
 newobject\_fromboxtext  
     Objects, 572  
 newobject\_fromdictionary  
     Objects, 572  
 newobject\_sprintf  
     Objects, 573  
 OBEX\_UTIL\_ATOM\_GETTEXT\_COMMAS\_DELIM  
     Atoms, 410  
 OBEX\_UTIL\_ATOM\_GETTEXT\_DEFAULT  
     Atoms, 410  
 OBEX\_UTIL\_ATOM\_GETTEXT\_FORCE\_ZEROS  
     Atoms, 410  
 OBEX\_UTIL\_ATOM\_GETTEXT\_LINEBREAK\_NODELIM  
     Atoms, 410  
 OBEX\_UTIL\_ATOM\_GETTEXT\_NOESCAPE  
     Atoms, 410  
 OBEX\_UTIL\_ATOM\_GETTEXT\_NUM\_HI\_RES  
     Atoms, 410  
 OBEX\_UTIL\_ATOM\_GETTEXT\_NUM\_LO\_RES  
     Atoms, 410  
 OBEX\_UTIL\_ATOM\_GETTEXT\_SYM\_FORCE\_QUOTE  
     Atoms, 410

OBEX\_UTIL\_ATOM\_GETTEXT\_SYM\_NO\_QUOTE  
Atoms, 410

OBEX\_UTIL\_ATOM\_GETTEXT\_TRUNCATE\_ZEROS  
Atoms, 410

OBJ\_ATTR\_ATOM  
Attributes, 196

OBJ\_ATTR\_ATOM\_ARRAY  
Attributes, 196

OBJ\_ATTR\_CHAR  
Attributes, 197

OBJ\_ATTR\_CHAR\_ARRAY  
Attributes, 197

OBJ\_ATTR\_DEFAULT  
Attributes, 198

OBJ\_ATTR\_DEFAULT\_SAVE  
Attributes, 198

OBJ\_ATTR\_DOUBLE  
Attributes, 198

OBJ\_ATTR\_DOUBLE\_ARRAY  
Attributes, 199

OBJ\_ATTR\_FLOAT  
Attributes, 199

OBJ\_ATTR\_FLOAT\_ARRAY  
Attributes, 200

OBJ\_ATTR\_LONG  
Attributes, 200

OBJ\_ATTR\_LONG\_ARRAY  
Attributes, 201

OBJ\_ATTR\_OBJ  
Attributes, 201

OBJ\_ATTR\_OBJ\_ARRAY  
Attributes, 202

OBJ\_ATTR\_SAVE  
Attributes, 202

OBJ\_ATTR\_SYM  
Attributes, 202

OBJ\_ATTR\_SYM\_ARRAY  
Attributes, 203

OBJ\_FLAG\_CLONE  
Data Storage, 280

OBJ\_FLAG\_DANGER  
Data Storage, 280

OBJ\_FLAG\_DATA  
Data Storage, 280

OBJ\_FLAG\_DEBUG  
Data Storage, 280

OBJ\_FLAG\_INHERITABLE  
Data Storage, 280

OBJ\_FLAG\_ITERATING  
Data Storage, 280

OBJ\_FLAG\_MEMORY  
Data Storage, 280

OBJ\_FLAG\_OBJ  
Data Storage, 280

OBJ\_FLAG\_REF  
Data Storage, 280

OBJ\_FLAG\_SILENT  
Data Storage, 280

Object Module, 911

jit\_object\_attach, 913

jit\_object\_attr\_get, 913

jit\_object\_attr\_usercanget, 914

jit\_object\_attr\_usercanset, 915

jit\_object\_class, 915

jit\_object\_classname, 916

jit\_object\_classname\_compare, 917

jit\_object\_detach, 918

jit\_object\_exportattrs, 918

jit\_object\_exportsummary, 919

jit\_object\_findregistered, 920

jit\_object\_findregisteredbyptr, 921

jit\_object\_free, 921

jit\_object\_getmethod, 922

jit\_object\_importattrs, 923

jit\_object\_method\_argsafe\_get, 924

jit\_object\_method\_imp, 925

jit\_object\_method\_typed, 926

jit\_object\_new\_imp, 927

jit\_object\_notify, 927

jit\_object\_register, 928

jit\_object\_unregister, 929

object\_addattr  
Attributes, 226

object\_alloc  
Objects, 574

object\_attach  
Objects, 575

object\_attach\_byptr  
Objects, 575

object\_attach\_byptr\_register  
Objects, 576

object\_attr\_get  
Attributes, 226

object\_attr\_get\_rect  
Attributes, 227

object\_attr\_getchar\_array  
Attributes, 227

object\_attr\_getcolor  
Attributes, 228

object\_attr\_getdouble\_array  
Attributes, 229

object\_attr\_getdump  
Attributes, 229

object\_attr\_getfill  
Styles, 813

object\_attr\_getfillcolor\_atposition  
Styles, 814

object\_attr\_getfloat

Attributes, 230  
`object_attr_getfloat_array`  
     Attributes, 230  
`object_attr_getjrgba`  
     Attributes, 231  
`object_attr_getlong`  
     Attributes, 231  
`object_attr_getlong_array`  
     Attributes, 232  
`object_attr_getpt`  
     Attributes, 233  
`object_attr_getsize`  
     Attributes, 233  
`object_attr_getsym`  
     Attributes, 234  
`object_attr_getsym_array`  
     Attributes, 234  
`object_attr_method`  
     Attributes, 235  
`object_attr_set_rect`  
     Attributes, 235  
`object_attr_set_xywh`  
     Attributes, 236  
`object_attr_setchar_array`  
     Attributes, 236  
`object_attr_setcolor`  
     Attributes, 237  
`object_attr_setdouble_array`  
     Attributes, 237  
`object_attr_setfloat`  
     Attributes, 238  
`object_attr_setfloat_array`  
     Attributes, 239  
`object_attr_setjrgba`  
     Attributes, 239  
`object_attr_setlong`  
     Attributes, 240  
`object_attr_setlong_array`  
     Attributes, 240  
`object_attr_setparse`  
     Attributes, 241  
`object_attr_setpt`  
     Attributes, 241  
`object_attr_setsize`  
     Attributes, 242  
`object_attr_setsym`  
     Attributes, 242  
`object_attr_setsym_array`  
     Attributes, 243  
`object_attr_setvalueof`  
     Attributes, 243  
`object_attr_touch`  
     Objects, 577  
`object_attr_touch_parse`  
       
     Objects, 577  
`object_attr_usercanget`  
     Attributes, 244  
`object_attr_usercanset`  
     Attributes, 244  
`object_chuckattr`  
     Attributes, 245  
`object_class`  
     Objects, 578  
`object_classname`  
     Objects, 578  
`object_classname_compare`  
     Objects, 579  
`object_deleteattr`  
     Attributes, 245  
`object_detach`  
     Objects, 579  
`object_detach_byptr`  
     Objects, 580  
`object_dictionaryarg`  
     Objects, 580  
`object_error`  
     Console, 518  
`object_error_obtrusive`  
     Console, 519  
`object_findregistered`  
     Objects, 581  
`object_findregisteredbyptr`  
     Objects, 581  
`object_free`  
     Objects, 582  
`object_getmethod`  
     Objects, 582  
`object_getvalueof`  
     Objects, 583  
`object_method`  
     Objects, 584  
`object_method_char`  
     Objects, 584  
`object_method_char_array`  
     Objects, 585  
`object_method_direct`  
     Objects, 570  
`object_method_double`  
     Objects, 585  
`object_method_double_array`  
     Objects, 586  
`object_method_float`  
     Objects, 587  
`object_method_float_array`  
     Objects, 587  
`object_method_format`  
     Objects, 588  
`object_method_long`

Objects, 588  
object\_method\_long\_array  
    Objects, 589  
object\_method\_obj  
    Objects, 590  
object\_method\_obj\_array  
    Objects, 590  
object\_method\_parse  
    Objects, 591  
object\_method\_sym  
    Objects, 591  
object\_method\_sym\_array  
    Objects, 592  
object\_method\_typed  
    Objects, 593  
object\_method\_typedfun  
    Objects, 593  
object\_new  
    Objects, 594  
object\_new\_parse  
    Attributes, 246  
object\_new\_typed  
    Objects, 595  
object\_notify  
    Objects, 595  
object\_obex\_dumpout  
    Objects, 596  
object\_obex\_lookup  
    Objects, 597  
object\_obex\_quicref  
    Miscellaneous, 511  
object\_obex\_store  
    Objects, 598  
object\_openhelp  
    Objects, 598  
object\_openquery  
    Objects, 598  
object\_openrefpage  
    Objects, 599  
object\_post  
    Console, 519  
object\_register  
    Objects, 599  
object\_register\_getnames  
    Objects, 600  
object\_setvalueof  
    Objects, 600  
object\_subscribe  
    Objects, 601  
object\_super\_method  
    Objects, 602  
object\_this\_method  
    Objects, 602  
object\_unregister  
    Objects, 603  
object\_unsubscribe  
    Objects, 603  
object\_warn  
    Console, 520  
Objects, 567  
    classname\_openhelp, 571  
    classname\_openquery, 571  
    classname\_openrefpage, 572  
    newobject\_fromboxtext, 572  
    newobject\_fromdictionary, 572  
    newobject\_sprintf, 573  
    object\_alloc, 574  
    object\_attach, 575  
    object\_attach\_byptr, 575  
    object\_attach\_byptr\_register, 576  
    object\_attr\_touch, 577  
    object\_attr\_touch\_parse, 577  
    object\_class, 578  
    object\_classname, 578  
    object\_classname\_compare, 579  
    object\_detach, 579  
    object\_detach\_byptr, 580  
    object\_dictionaryarg, 580  
    object\_findregistered, 581  
    object\_findregisteredbyptr, 581  
    object\_free, 582  
    object\_getmethod, 582  
    object\_getvalueof, 583  
    object\_method, 584  
    object\_method\_char, 584  
    object\_method\_char\_array, 585  
    object\_method\_direct, 570  
    object\_method\_double, 585  
    object\_method\_double\_array, 586  
    object\_method\_float, 587  
    object\_method\_float\_array, 587  
    object\_method\_format, 588  
    object\_method\_long, 588  
    object\_method\_long\_array, 589  
    object\_method\_obj, 590  
    object\_method\_obj\_array, 590  
    object\_method\_parse, 591  
    object\_method\_sym, 591  
    object\_method\_sym\_array, 592  
    object\_method\_typed, 593  
    object\_method\_typedfun, 593  
    object\_new, 594  
    object\_new\_typed, 595  
    object\_notify, 595  
    object\_obex\_dumpout, 596  
    object\_obex\_lookup, 597  
    object\_obex\_store, 598  
    object\_openhelp, 598

object\_openquery, 598  
 object\_openrefpage, 599  
 object\_register, 599  
 object\_register\_getnames, 600  
 object\_setvalueof, 600  
 object\_subscribe, 601  
 object\_super\_method, 602  
 object\_this\_method, 602  
 object\_unregister, 603  
 object\_unsubscribe, 603  
**Old-Style Classes**, 259  
 addbang, 260  
 addfloat, 260  
 addftx, 260  
 addint, 261  
 addinx, 261  
 addmess, 262  
 alias, 262  
 class\_setname, 262  
 egetfn, 263  
 freeobject, 263  
 getfn, 264  
 newinstance, 264  
 newobject, 265  
 setup, 265  
 typedmess, 267  
 zgetfn, 268  
**open\_dialog**  
 Files and Folders, 458  
**open\_promptset**  
 Files and Folders, 459  
**Operator Vector Module**, 1082  
 jit\_op\_vector\_abs\_float32, 1090  
 jit\_op\_vector\_abs\_float64, 1090  
 jit\_op\_vector\_abs\_long, 1091  
 jit\_op\_vector\_absdiff\_char, 1091  
 jit\_op\_vector\_absdiff\_float32, 1092  
 jit\_op\_vector\_absdiff\_float64, 1092  
 jit\_op\_vector\_absdiff\_long, 1093  
 jit\_op\_vector\_acos\_float32, 1093  
 jit\_op\_vector\_acos\_float64, 1094  
 jit\_op\_vector\_acosh\_float32, 1094  
 jit\_op\_vector\_acosh\_float64, 1094  
 jit\_op\_vector\_add\_char, 1095  
 jit\_op\_vector\_add\_float32, 1095  
 jit\_op\_vector\_add\_float64, 1096  
 jit\_op\_vector\_add\_long, 1096  
 jit\_op\_vector\_adds\_char, 1097  
 jit\_op\_vector\_and\_char, 1097  
 jit\_op\_vector\_and\_float32, 1098  
 jit\_op\_vector\_and\_float64, 1098  
 jit\_op\_vector\_and\_long, 1099  
 jit\_op\_vector\_asin\_float32, 1099  
 jit\_op\_vector\_asin\_float64, 1099  
 jit\_op\_vector\_asinh\_float32, 1100  
 jit\_op\_vector\_asinh\_float64, 1100  
 jit\_op\_vector\_atan2\_float32, 1101  
 jit\_op\_vector\_atan2\_float64, 1101  
 jit\_op\_vector\_atan\_float32, 1102  
 jit\_op\_vector\_atan\_float64, 1102  
 jit\_op\_vector\_atanh\_float32, 1103  
 jit\_op\_vector\_atanh\_float64, 1103  
 jit\_op\_vector\_avg\_char, 1104  
 jit\_op\_vector\_avg\_float32, 1104  
 jit\_op\_vector\_avg\_float64, 1104  
 jit\_op\_vector\_avg\_long, 1105  
 jit\_op\_vector\_bitand\_char, 1105  
 jit\_op\_vector\_bitand\_long, 1106  
 jit\_op\_vector\_bitnot\_char, 1106  
 jit\_op\_vector\_bitnot\_long, 1107  
 jit\_op\_vector\_bitor\_char, 1107  
 jit\_op\_vector\_bitor\_long, 1108  
 jit\_op\_vector\_bitxor\_char, 1108  
 jit\_op\_vector\_bitxor\_long, 1109  
 jit\_op\_vector\_ceil\_float32, 1109  
 jit\_op\_vector\_ceil\_float64, 1109  
 jit\_op\_vector\_cos\_float32, 1110  
 jit\_op\_vector\_cos\_float64, 1110  
 jit\_op\_vector\_cosh\_float32, 1111  
 jit\_op\_vector\_cosh\_float64, 1111  
 jit\_op\_vector\_div\_char, 1112  
 jit\_op\_vector\_div\_float32, 1112  
 jit\_op\_vector\_div\_float64, 1113  
 jit\_op\_vector\_div\_long, 1113  
 jit\_op\_vector\_eq\_char, 1114  
 jit\_op\_vector\_eq\_float32, 1114  
 jit\_op\_vector\_eq\_float64, 1115  
 jit\_op\_vector\_eq\_long, 1115  
 jit\_op\_vector\_eqp\_char, 1116  
 jit\_op\_vector\_eqp\_float32, 1116  
 jit\_op\_vector\_eqp\_float64, 1117  
 jit\_op\_vector\_eqp\_long, 1117  
 jit\_op\_vector\_exp2\_float32, 1117  
 jit\_op\_vector\_exp2\_float64, 1118  
 jit\_op\_vector\_exp\_float32, 1118  
 jit\_op\_vector\_exp\_float64, 1119  
 jit\_op\_vector\_flipdiv\_char, 1119  
 jit\_op\_vector\_flipdiv\_float32, 1120  
 jit\_op\_vector\_flipdiv\_float64, 1121  
 jit\_op\_vector\_flipdiv\_long, 1121  
 jit\_op\_vector\_flipmod\_char, 1122  
 jit\_op\_vector\_flipmod\_float32, 1123  
 jit\_op\_vector\_flipmod\_float64, 1124  
 jit\_op\_vector\_flipmod\_long, 1124  
 jit\_op\_vector\_flippass\_char, 1125  
 jit\_op\_vector\_flippass\_float32, 1126  
 jit\_op\_vector\_flippass\_float64, 1127  
 jit\_op\_vector\_flippass\_long, 1127

jit\_op\_vector\_flipsub\_char, 1128  
jit\_op\_vector\_flipsub\_float32, 1129  
jit\_op\_vector\_flipsub\_long, 1130  
jit\_op\_vector\_floor\_float32, 1130  
jit\_op\_vector\_floor\_float64, 1131  
jit\_op\_vector\_fold\_float32, 1131  
jit\_op\_vector\_fold\_float64, 1132  
jit\_op\_vector\_gt\_char, 1132  
jit\_op\_vector\_gt\_float32, 1133  
jit\_op\_vector\_gt\_float64, 1133  
jit\_op\_vector\_gt\_long, 1133  
jit\_op\_vector\_gte\_char, 1134  
jit\_op\_vector\_gte\_float32, 1134  
jit\_op\_vector\_gte\_float64, 1135  
jit\_op\_vector\_gte\_long, 1135  
jit\_op\_vector\_gtep\_char, 1136  
jit\_op\_vector\_gtep\_float32, 1136  
jit\_op\_vector\_gtep\_float64, 1137  
jit\_op\_vector\_gtep\_long, 1137  
jit\_op\_vector\_gtp\_char, 1138  
jit\_op\_vector\_gtp\_float32, 1138  
jit\_op\_vector\_gtp\_float64, 1138  
jit\_op\_vector\_gtp\_long, 1139  
jit\_op\_vector\_hypot\_float32, 1139  
jit\_op\_vector\_hypot\_float64, 1140  
jit\_op\_vector\_log10\_float32, 1140  
jit\_op\_vector\_log10\_float64, 1141  
jit\_op\_vector\_log2\_float32, 1141  
jit\_op\_vector\_log2\_float64, 1142  
jit\_op\_vector\_log\_float32, 1142  
jit\_op\_vector\_log\_float64, 1143  
jit\_op\_vector\_lshift\_char, 1143  
jit\_op\_vector\_lshift\_long, 1143  
jit\_op\_vector\_lt\_char, 1144  
jit\_op\_vector\_lt\_float32, 1144  
jit\_op\_vector\_lt\_float64, 1145  
jit\_op\_vector\_lt\_long, 1145  
jit\_op\_vector\_lte\_char, 1146  
jit\_op\_vector\_lte\_float32, 1146  
jit\_op\_vector\_lte\_float64, 1147  
jit\_op\_vector\_lte\_long, 1147  
jit\_op\_vector\_ltep\_char, 1148  
jit\_op\_vector\_ltep\_float32, 1148  
jit\_op\_vector\_ltep\_float64, 1148  
jit\_op\_vector\_ltep\_long, 1149  
jit\_op\_vector\_ltp\_char, 1149  
jit\_op\_vector\_ltp\_float32, 1150  
jit\_op\_vector\_ltp\_float64, 1150  
jit\_op\_vector\_ltp\_long, 1151  
jit\_op\_vector\_max\_char, 1151  
jit\_op\_vector\_max\_float32, 1152  
jit\_op\_vector\_max\_float64, 1152  
jit\_op\_vector\_max\_long, 1153  
jit\_op\_vector\_min\_char, 1153  
jit\_op\_vector\_min\_float32, 1153  
jit\_op\_vector\_min\_float64, 1154  
jit\_op\_vector\_min\_long, 1154  
jit\_op\_vector\_mod\_char, 1155  
jit\_op\_vector\_mod\_float32, 1155  
jit\_op\_vector\_mod\_float64, 1156  
jit\_op\_vector\_mod\_long, 1156  
jit\_op\_vector\_mult\_char, 1157  
jit\_op\_vector\_mult\_float32, 1157  
jit\_op\_vector\_mult\_float64, 1158  
jit\_op\_vector\_mult\_long, 1158  
jit\_op\_vector\_neq\_char, 1159  
jit\_op\_vector\_neq\_float32, 1159  
jit\_op\_vector\_neq\_float64, 1160  
jit\_op\_vector\_neq\_long, 1160  
jit\_op\_vector\_neqp\_char, 1160  
jit\_op\_vector\_neqp\_float32, 1161  
jit\_op\_vector\_neqp\_float64, 1161  
jit\_op\_vector\_neqp\_long, 1162  
jit\_op\_vector\_not\_char, 1162  
jit\_op\_vector\_not\_float32, 1163  
jit\_op\_vector\_not\_float64, 1163  
jit\_op\_vector\_not\_long, 1164  
jit\_op\_vector\_or\_char, 1164  
jit\_op\_vector\_or\_float32, 1165  
jit\_op\_vector\_or\_float64, 1165  
jit\_op\_vector\_or\_long, 1165  
jit\_op\_vector\_pass\_char, 1166  
jit\_op\_vector\_pass\_float32, 1166  
jit\_op\_vector\_pass\_float64, 1167  
jit\_op\_vector\_pass\_long, 1167  
jit\_op\_vector\_pow\_float32, 1168  
jit\_op\_vector\_pow\_float64, 1168  
jit\_op\_vector\_round\_float32, 1169  
jit\_op\_vector\_round\_float64, 1169  
jit\_op\_vector\_rshift\_char, 1170  
jit\_op\_vector\_rshift\_long, 1170  
jit\_op\_vector\_sin\_float32, 1171  
jit\_op\_vector\_sin\_float64, 1171  
jit\_op\_vector\_sinh\_float32, 1172  
jit\_op\_vector\_sinh\_float64, 1172  
jit\_op\_vector\_sqrt\_float32, 1172  
jit\_op\_vector\_sqrt\_float64, 1173  
jit\_op\_vector\_sub\_char, 1173  
jit\_op\_vector\_sub\_float32, 1174  
jit\_op\_vector\_sub\_float64, 1174  
jit\_op\_vector\_sub\_long, 1175  
jit\_op\_vector\_subs\_char, 1175  
jit\_op\_vector\_tan\_float32, 1176  
jit\_op\_vector\_tan\_float64, 1176  
jit\_op\_vector\_tanh\_float32, 1177  
jit\_op\_vector\_tanh\_float64, 1177  
jit\_op\_vector\_trunc\_float32, 1178  
jit\_op\_vector\_trunc\_float64, 1178

jit\_op\_vector\_wrap\_float32, 1179  
 jit\_op\_vector\_wrap\_float64, 1179  
**ouchstring**  
 Console, 520  
**outlet\_anything**  
 Inlets and Outlets, 273  
**outlet\_bang**  
 Inlets and Outlets, 274  
**outlet\_float**  
 Inlets and Outlets, 274  
**outlet\_int**  
 Inlets and Outlets, 274  
**outlet\_list**  
 Inlets and Outlets, 275  
**outlet\_new**  
 Inlets and Outlets, 276  
  
**Parallel Utility Module**, 1055  
 jit\_parallel\_ndim\_calc, 1055  
 jit\_parallel\_ndim\_simplecalc1, 1057  
 jit\_parallel\_ndim\_simplecalc2, 1058  
 jit\_parallel\_ndim\_simplecalc3, 1059  
 jit\_parallel\_ndim\_simplecalc4, 1060  
**Patcher**, 604  
 PI\_DEEP, 606  
 PI\_REQUIREFIRSTIN, 606  
 PI\_WANTBOX, 606  
 t\_box, 605  
 t\_patcher, 606  
**patcherview\_canvas\_to\_screen**  
 jpatcherview, 667  
**patcherview\_findpatcherview**  
 jpatcherview, 667  
**patcherview\_get\_jgraphics**  
 jpatcherview, 668  
**patcherview\_get\_locked**  
 jpatcherview, 668  
**patcherview\_get\_nextview**  
 jpatcherview, 668  
**patcherview\_get\_patcher**  
 jpatcherview, 670  
**patcherview\_get\_presentation**  
 jpatcherview, 670  
**patcherview\_get\_rect**  
 jpatcherview, 670  
**patcherview\_get\_topview**  
 jpatcherview, 671  
**patcherview\_get\_visible**  
 jpatcherview, 671  
**patcherview\_get\_zoomfactor**  
 jpatcherview, 672  
**patcherview\_screen\_to\_canvas**  
 jpatcherview, 672  
**patcherview\_set\_locked**

jpatcherview, 673  
**patcherview\_set\_presentation**  
 jpatcherview, 673  
**patcherview\_set\_rect**  
 jpatcherview, 673  
**patcherview\_set\_visible**  
 jpatcherview, 674  
**patcherview\_set\_zoomfactor**  
 jpatcherview, 674  
**path\_absolutepath**  
 Files and Folders, 459  
**path\_closefolder**  
 Files and Folders, 460  
**path\_createsysfile**  
 Files and Folders, 460  
**path\_exists**  
 Files and Folders, 461  
**path\_fileinfo**  
 Files and Folders, 461  
**PATH\_FILEINFO\_ALIAS**  
 Files and Folders, 450  
**PATH\_FILEINFO\_FOLDER**  
 Files and Folders, 450  
**PATH\_FILEINFO\_PACKAGE**  
 Files and Folders, 450  
**PATH\_FOLDER\_SNIFF**  
 Files and Folders, 452  
**path\_foldernextfile**  
 Files and Folders, 462  
**path\_frompathname**  
 Files and Folders, 463  
**path\_getapppath**  
 Files and Folders, 463  
**path\_getdefault**  
 Files and Folders, 463  
**path\_getfilemoddate**  
 Files and Folders, 464  
**path\_getmoddate**  
 Files and Folders, 464  
**path\_nameconform**  
 Files and Folders, 465  
**PATH\_NOALIASRESOLUTION**  
 Files and Folders, 452  
**path\_openfolder**  
 Files and Folders, 465  
**path\_opensysfile**  
 Files and Folders, 466  
**PATH\_READ\_PERM**  
 Files and Folders, 450  
**PATH\_REPORTPACKAGEASFOLDER**  
 Files and Folders, 452  
**path\_resolvefile**  
 Files and Folders, 466  
**PATH\_RW\_PERM**

Files and Folders, 450  
path\_setdefault  
Files and Folders, 467  
PATH\_STYLE\_COLON  
Files and Folders, 452  
PATH\_STYLE\_MAX  
Files and Folders, 452  
PATH\_STYLE\_NATIVE  
Files and Folders, 452  
PATH\_STYLE\_NATIVE\_WIN  
Files and Folders, 452  
PATH\_STYLE\_SLASH  
Files and Folders, 452  
path\_toabsolutestystempath  
Files and Folders, 467  
path\_topathname  
Files and Folders, 468  
path\_topotentialname  
Files and Folders, 468  
PATH\_TYPE\_ABSOLUTE  
Files and Folders, 453  
PATH\_TYPE\_BOOT  
Files and Folders, 453  
PATH\_TYPE\_C74  
Files and Folders, 453  
PATH\_TYPE\_DESKTOP  
Files and Folders, 453  
PATH\_TYPE\_IGNORE  
Files and Folders, 452  
PATH\_TYPE\_MAXDB  
Files and Folders, 453  
PATH\_TYPE\_PATH  
Files and Folders, 453  
PATH\_TYPE\_PLUGIN  
Files and Folders, 453  
PATH\_TYPE\_RELATIVE  
Files and Folders, 453  
PATH\_TYPE\_TEMPFOLDER  
Files and Folders, 453  
PATH\_TYPE\_TILDE  
Files and Folders, 453  
PATH\_TYPE\_USERMAX  
Files and Folders, 453  
PATH\_WRITE\_PERM  
Files and Folders, 450  
PFFT, 565  
PI  
MSP, 549  
PI\_DEEP  
Patcher, 606  
PI\_REQUIREFIRSTIN  
Patcher, 606  
PI\_WANTBOX  
Patcher, 606

PIOVERTWO  
MSP, 549  
Poly, 566  
Popup Menus, 835  
jpopupmenu\_additem, 836  
jpopupmenu\_addseparator, 836  
jpopupmenu\_addsubmenu, 837  
jpopupmenu\_clear, 837  
jpopupmenu\_create, 837  
jpopupmenu\_destroy, 838  
jpopupmenu\_popup, 838  
jpopupmenu\_popup\_abovebox, 838  
jpopupmenu\_popup\_aboverect, 839  
jpopupmenu\_popup\_belowrect, 839  
jpopupmenu\_popup\_nearbox, 840  
jpopupmenu\_popup\_nearbox\_with\_options, 840  
jpopupmenu\_setcolors, 841  
jpopupmenu\_setfont, 841

post  
Console, 521  
post\_sym  
Miscellaneous, 511  
postargs  
Atoms, 433  
postatom  
Console, 522  
postdictionary  
Dictionary, 339  
preset\_int  
Presets, 533  
preset\_set  
Presets, 533  
preset\_store  
Presets, 533  
Presets, 531  
preset\_int, 533  
preset\_set, 533  
preset\_store, 533  
proxy\_getinlet  
Inlets and Outlets, 276  
proxy\_new  
Inlets and Outlets, 277

qelem\_free  
Qelems, 689  
qelem\_front  
Qelems, 690  
qelem\_new  
Qelems, 690  
qelem\_set  
Qelems, 690  
qelem\_unset  
Qelems, 691  
Qelems, 688

qelem\_free, 689  
 qelem\_front, 690  
 qelem\_new, 690  
 qelem\_set, 690  
 qelem\_unset, 691  
**Quick Map**, 385  
 quickmap\_add, 386  
 quickmap\_drop, 387  
 quickmap\_lookup\_key1, 387  
 quickmap\_lookup\_key2, 388  
 quickmap\_new, 388  
 quickmap\_READONLY, 388  
 quickmap\_ADD  
     Quick Map, 386  
 quickmap\_DROP  
     Quick Map, 387  
 quickmap\_LOOKUP\_KEY1  
     Quick Map, 387  
 quickmap\_LOOKUP\_KEY2  
     Quick Map, 388  
 quickmap\_NEW  
     Quick Map, 388  
 quickmap\_READONLY  
     Quick Map, 388  
 quittask\_install  
     Miscellaneous, 512  
 quittask\_remove  
     Miscellaneous, 512  
**readatom**  
 Binbufs, 441  
**readtohandle**  
 Loading Max Files, 536  
**saveas\_dialog**  
 Files and Folders, 469  
**saveas\_promptset**  
 Files and Folders, 470  
**saveasdialoG\_EXTENDED**  
 Files and Folders, 470  
**Scalable Vector Graphics**, 778  
 jsvg\_create\_from\_file, 778  
 jsvg\_create\_from\_resource, 779  
 jsvg\_create\_from\_xmlstring, 779  
 jsvg\_destroy, 780  
 jsvg\_get\_size, 780  
 jsvg\_render, 780  
**schedule**  
 Threads, 721  
**schedule\_delay**  
 Threads, 722  
**scheduler\_fromobject**  
 Clocks, 682  
**scheduler\_get**  
 Clocks, 682  
 scheduler\_gettime  
     Clocks, 683  
**scheduler\_new**  
     Clocks, 683  
**scheduler\_run**  
     Clocks, 684  
**scheduler\_set**  
     Clocks, 684  
**scheduler\_settime**  
     Clocks, 684  
**scheduler\_shift**  
     Clocks, 685  
**setclock\_delay**  
     Clocks, 685  
**setclock\_fdelay**  
     Clocks, 686  
**setclock\_gettime**  
     Clocks, 686  
**setclock\_gettime**  
     Clocks, 687  
**setclock\_unset**  
     Clocks, 687  
**setup**  
     Old-Style Classes, 265  
**snprintf\_zero**  
     Miscellaneous, 512  
**String Object**, 389  
 string\_append, 390  
 string\_chop, 390  
 string\_getptr, 390  
 string\_new, 391  
 string\_reserve, 391  
**string\_append**  
     String Object, 390  
**string\_chop**  
     String Object, 390  
**string\_getptr**  
     String Object, 390  
**string\_new**  
     String Object, 391  
**string\_reserve**  
     String Object, 391  
**stringload**  
 Loading Max Files, 536  
**strncat\_zero**  
     Miscellaneous, 513  
**strncpy\_zero**  
     Miscellaneous, 513  
**STRUCT\_ATTR\_ATOM**  
     Attributes, 203  
**STRUCT\_ATTR\_ATOM\_ARRAY**  
     Attributes, 204  
**STRUCT\_ATTR\_ATOM\_LONG**  
     Attributes, 204

STRUCT\_ATTR\_ATOM\_VARSIZE  
    Attributes, 205

STRUCT\_ATTR\_CHAR  
    Attributes, 205

STRUCT\_ATTR\_CHAR\_ARRAY  
    Attributes, 206

STRUCT\_ATTR\_CHAR\_VARSIZE  
    Attributes, 206

STRUCT\_ATTR\_DOUBLE  
    Attributes, 207

STRUCT\_ATTR\_DOUBLE\_ARRAY  
    Attributes, 207

STRUCT\_ATTR\_DOUBLE\_VARSIZE  
    Attributes, 208

STRUCT\_ATTR\_FLOAT  
    Attributes, 208

STRUCT\_ATTR\_FLOAT\_ARRAY  
    Attributes, 209

STRUCT\_ATTR\_FLOAT\_VARSIZE  
    Attributes, 209

STRUCT\_ATTR\_LONG  
    Attributes, 210

STRUCT\_ATTR\_LONG\_ARRAY  
    Attributes, 210

STRUCT\_ATTR\_LONG\_VARSIZE  
    Attributes, 211

STRUCT\_ATTR\_OBJ  
    Attributes, 212

STRUCT\_ATTR\_OBJ\_ARRAY  
    Attributes, 212

STRUCT\_ATTR\_OBJ\_VARSIZE  
    Attributes, 212

STRUCT\_ATTR\_SYM  
    Attributes, 213

STRUCT\_ATTR\_SYM\_ARRAY  
    Attributes, 213

STRUCT\_ATTR\_SYM\_VARSIZE  
    Attributes, 214

structmembersize  
    Miscellaneous, 506

Styles, 805

- class\_attr\_setfill, 810
- class\_attr\_setstyle, 810
- class\_attr\_style\_alias, 811
- CLASS\_ATTR\_STYLE\_ALIAS\_COMPATIBILITY, 806
- CLASS\_ATTR\_STYLE\_ALIAS\_NOSAVE, 807
- CLASS\_ATTR\_STYLE\_ALIAS\_RGBA\_LEGACY, 807
- CLASS\_ATTR\_STYLE\_RGBA, 808
- CLASS\_ATTR\_STYLE\_RGBA\_NOSAVE, 808
- CLASS\_ATTR\_STYLE\_RGBA\_PREVIEW, 809
- class\_attr\_stylemap, 811
- FILL\_ATTR\_SAVE, 810

jgraphics\_attr\_fillrect, 812

jgraphics\_attr\_setfill, 813

object\_attr\_getfill, 813

object\_attr\_getfillcolor\_atposition, 814

swapf32  
    Miscellaneous Utility Module, 935

swapf64  
    Miscellaneous Utility Module, 936

Symbol Object, 391

- symobject\_linklist\_match, 392
- symobject\_new, 393

symbol\_stripquotes  
    Miscellaneous, 514

symbol\_unique  
    Miscellaneous, 514

symbolarray\_sort  
    Miscellaneous, 514

Symbols, 442

- gensym, 443
- gensym\_tr, 444

symobject\_linklist\_match  
    Symbol Object, 392

symobject\_new  
    Symbol Object, 393

sys\_getblksize  
    MSP, 553

sys\_getdspobjdspstate  
    MSP, 553

sys\_getdspstate  
    MSP, 554

sys\_getmaxblksize  
    MSP, 554

sys\_getsr  
    MSP, 554

SYS\_MAXBLKSIZE  
    MSP, 551

SYSDATEFORMAT\_FLAGS\_LONG  
    Systime API, 693

SYSDATEFORMAT\_FLAGS\_MEDIUM  
    Systime API, 693

SYSDATEFORMAT\_FLAGS\_SHORT  
    Systime API, 693

sysdateformat\_formatdatetime  
    Systime API, 693

sysdateformat\_strtimetodatetime  
    Systime API, 694

SYSFILE\_ATMARK  
    Files and Folders, 453

sysfile\_close  
    Files and Folders, 471

SYSFILE\_FROMLEOF  
    Files and Folders, 453

SYSFILE\_FROMMARK  
    Files and Folders, 453

SYSFILE\_FROMSTART  
     Files and Folders, 453

sysfile\_geteof  
     Files and Folders, 472

sysfile\_getpos  
     Files and Folders, 472

sysfile\_openhandle  
     Files and Folders, 473

sysfile\_opendirsize  
     Files and Folders, 473

sysfile\_read  
     Files and Folders, 474

sysfile\_readtextfile  
     Files and Folders, 474

sysfile\_readtobuffer  
     Files and Folders, 475

sysfile\_readtopt  
     Files and Folders, 475

sysfile\_seteof  
     Files and Folders, 476

sysfile\_setpos  
     Files and Folders, 476

sysfile\_spoolcopy  
     Files and Folders, 477

sysfile\_write  
     Files and Folders, 477

sysfile\_writetextfile  
     Files and Folders, 478

sysmem\_copyptr  
     Memory Management, 493

sysmem\_freehandle  
     Memory Management, 493

sysmem\_freeptr  
     Memory Management, 493

sysmem\_handlesize  
     Memory Management, 494

sysmem\_lockhandle  
     Memory Management, 494

sysmem\_newhandle  
     Memory Management, 495

sysmem\_newhandleclear  
     Memory Management, 495

sysmem\_newptr  
     Memory Management, 496

sysmem\_newptrclear  
     Memory Management, 496

sysmem\_nulterminatehandle  
     Memory Management, 497

sysmem\_ptrrandhand  
     Memory Management, 497

sysmem\_ptrbeforehand  
     Memory Management, 497

sysmem\_ptrsize  
     Memory Management, 498

sysmem\_resizehandle  
     Memory Management, 498

sysmem\_resizeptr  
     Memory Management, 499

sysmem\_resizeptrclear  
     Memory Management, 499

systemfontname  
     JFont, 794

systemfontname\_bold  
     JFont, 794

systemfontname\_light  
     JFont, 794

systemfontsym  
     JFont, 795

systhread\_create  
     Threads, 723

systhread\_detach  
     Threads, 723

systhread\_equal  
     Threads, 724

systhread\_exit  
     Threads, 724

systhread\_getpriority  
     Threads, 725

systhread\_isaudiothread  
     Threads, 725

systhread\_ismainthread  
     Threads, 725

systhread\_istimerthread  
     Threads, 725

systhread\_join  
     Threads, 726

SYSTHREAD\_MUTEX\_ERRORCHECK  
     Threads, 719

systhread\_mutex\_free  
     Mutexes, 733

systhread\_mutex\_lock  
     Mutexes, 733

systhread\_mutex\_new  
     Mutexes, 733

systhread\_mutex\_newlock  
     Mutexes, 734

SYSTHREAD\_MUTEX\_NORMAL  
     Threads, 719

SYSTHREAD\_MUTEX\_RECURSIVE  
     Threads, 719

systhread\_mutex\_trylock  
     Mutexes, 734

systhread\_mutex\_unlock  
     Mutexes, 735

systhread\_self  
     Threads, 726

systhread\_set\_name  
     Threads, 726

systhread\_setpriority  
    Threads, 727

systhread\_sleep  
    Threads, 727

systhread\_terminate  
    Threads, 727

Systime API, 691

- e\_max\_dateflags, 693
- SYSDATEFORMAT\_FLAGS\_LONG, 693
- SYSDATEFORMAT\_FLAGS\_MEDIUM, 693
- SYSDATEFORMAT\_FLAGS\_SHORT, 693
- sysdateformat\_formatdatetime, 693
- sysdateformat\_strtimetodatetime, 694
- systime\_datetime, 694
- systime\_datetime\_milliseconds, 694
- systime\_datetoseconds, 694
- systime\_ms, 695
- systime\_seconds, 695
- systime\_secondstodate, 695
- systime\_ticks, 696

systime\_datetime

- Systime API, 694

systime\_datetime\_milliseconds

- Systime API, 694

systime\_datetoseconds

- Systime API, 694

systime\_ms

- Systime API, 695

systime\_seconds

- Systime API, 695

systime\_secondstodate

- Systime API, 695

systime\_ticks

- Systime API, 696

system\_gettime

- Clocks, 688

t\_atom, 1181

t\_atomarray, 1182

t\_atomarray\_flags

- Atom Array, 282

t\_atombuf, 1182

t\_attr, 1183

t\_box

- Patcher, 605

t\_buffer, 1184

- b\_dirty\_clock, 1187

t\_buffer\_info, 1187

t\_buffer\_obj

- Buffers, 558

t\_buffer\_ref

- Buffers, 558

t\_celldesc, 1188

t\_charset\_converter, 1189

t\_class, 1189

t\_cmppfn

- Data Storage, 279

t\_database

- Database, 292

t\_datetime, 1190

t\_db\_result

- Database, 292

t\_db\_view

- Database, 292

t\_dictionary, 1191

t\_dictionary\_entry, 1192

t\_double

- MSP, 550

t\_ex\_ex, 1192

t\_expr, 1193

t\_filehandle

- Files and Folders, 449

t\_fileinfo, 1194

t\_float

- MSP, 550

t\_funbuff, 1194

t\_hashtab, 1196

t\_hashtab\_entry, 1197

t\_indexmap, 1197

t\_indexmap\_entry, 1198

t\_itm

- ITM Time Objects, 699

t\_jbox, 1199

t\_jboxdrawparams, 1200

t\_jcolumn, 1200

t\_jdataview, 1204

t\_jgraphics\_fileformat

- JGraphics, 742

t\_jgraphics\_font\_extents, 1207

t\_jgraphics\_font\_slant

- JFont, 783

t\_jgraphics\_font\_weight

- JFont, 783

t\_jgraphics\_format

- JGraphics, 742

t\_jgraphics\_text\_justification

- JGraphics, 744

t\_jgraphics\_textlayout\_flags

- TextLayout, 829

t\_jit\_attr, 1208

t\_jit\_attr\_filter\_clip, 1209

t\_jit\_attr\_filter\_proc, 1210

t\_jit\_attr\_offset, 1211

t\_jit\_attr\_offset\_array, 1212

t\_jit\_attribute, 1213

t\_jit\_gl\_buffer\_data, 1215

t\_jit\_gl\_buffer\_view, 1216

t\_jit\_gl\_context\_view, 1216

t\_jit\_gl\_drawinfo, 1219  
 t\_jit\_glchunk, 1219  
 t\_jit\_matrix\_info, 1220  
 t\_jit\_mop, 1221  
 t\_jit\_mop\_io, 1223  
 t\_jit\_op\_info, 1224  
 t\_jmatrix, 1225  
 t\_jmouse\_cursortype  
     Mouse and Keyboard, 543  
 t\_jrgb, 1225  
 t\_jrgba, 1226  
 t\_line\_3d, 1226  
 t\_linklist, 1227  
 t\_llelem, 1228  
 t\_matrix\_conv\_info, 1228  
 t\_messlist, 1229  
 t\_modifiers  
     Mouse and Keyboard, 544  
 t\_object, 1230  
 t\_package\_file, 1230  
 t\_patcher  
     Patcher, 606  
 t\_path, 1231  
 t\_pathlink, 1231  
 t\_perfroutine  
     MSP, 556  
 t\_pfftpub, 1232  
 t\_privatesortrec, 1233  
 t\_pt, 1234  
 t\_pxdata, 1234  
 t\_pxjbox, 1235  
 t\_pxobject, 1236  
 t\_quickmap, 1237  
 t\_rect, 1238  
 t\_sample  
     MSP, 550  
 t\_signal, 1238  
 t\_size, 1239  
 t\_string, 1239  
 t\_symbol, 1240  
 t\_symobject, 1241  
 t\_sysfile\_pos\_mode  
     Files and Folders, 453  
 t\_sysfile\_text\_flags  
     Files and Folders, 453  
 t\_timeobject  
     ITM Time Objects, 716  
 t\_tinyobject, 1241  
 t\_vptr  
     MSP, 550  
 t\_wind\_mouse\_info, 1242  
 t\_zll, 1243  
 Table Access, 529  
     table\_dirty, 529  
     table\_get, 530  
     Table Access, 529  
 table\_get  
     Table Access, 530  
 Text Editor Windows, 530  
 TEXT\_ENCODING\_USE\_FILE  
     Files and Folders, 453  
 TEXT\_LB\_MAC  
     Files and Folders, 453  
 TEXT\_LB\_NATIVE  
     Files and Folders, 453  
 TEXT\_LB\_PC  
     Files and Folders, 453  
 TEXT\_LB\_UNIX  
     Files and Folders, 453  
 TEXT\_NULL\_TERMINATE  
     Files and Folders, 453  
 TextField, 814  
     textfield\_get\_autoscroll, 816  
     textfield\_get\_bgcolor, 816  
     textfield\_get\_editonclick, 817  
     textfield\_get\_emptytext, 817  
     textfield\_get\_noactivate, 818  
     textfield\_get\_owner, 818  
     textfield\_get\_READONLY, 818  
     textfield\_get\_selectallonedit, 819  
     textfield\_get\_textcolor, 819  
     textfield\_get\_textmargins, 819  
     textfield\_get\_underline, 820  
     textfield\_get\_useellipsis, 820  
     textfield\_get\_wantsreturn, 821  
     textfield\_get\_wantstab, 821  
     textfield\_get\_wordwrap, 821  
     textfield\_set\_autoscroll, 822  
     textfield\_set\_bgcolor, 822  
     textfield\_set\_editonclick, 823  
     textfield\_set\_emptytext, 823  
     textfield\_set\_noactivate, 824  
     textfield\_set\_READONLY, 824  
     textfield\_set\_selectallonedit, 824  
     textfield\_set\_textcolor, 825  
     textfield\_set\_textmargins, 825  
     textfield\_set\_underline, 826  
     textfield\_set\_useellipsis, 826  
     textfield\_set\_wantsreturn, 827  
     textfield\_set\_wantstab, 827  
     textfield\_set\_wordwrap, 827  
 textfield\_get\_autoscroll  
     TextField, 816  
 textfield\_get\_bgcolor  
     TextField, 816  
 textfield\_get\_editonclick  
     TextField, 817

textfield\_get\_emptytext  
    TextField, 817  
textfield\_get\_noactivate  
    TextField, 818  
textfield\_get\_owner  
    TextField, 818  
textfield\_get\_READONLY  
    TextField, 818  
textfield\_get\_selectallonedit  
    TextField, 819  
textfield\_get\_textcolor  
    TextField, 819  
textfield\_get\_textmargins  
    TextField, 819  
textfield\_get\_underline  
    TextField, 820  
textfield\_get\_useellipsis  
    TextField, 820  
textfield\_get\_wantsreturn  
    TextField, 821  
textfield\_get\_wantstab  
    TextField, 821  
textfield\_get\_wordwrap  
    TextField, 821  
textfield\_set\_autoscroll  
    TextField, 822  
textfield\_set\_bgcolor  
    TextField, 822  
textfield\_set\_editonclick  
    TextField, 823  
textfield\_set\_emptytext  
    TextField, 823  
textfield\_set\_noactivate  
    TextField, 824  
textfield\_set\_READONLY  
    TextField, 824  
textfield\_set\_selectallonedit  
    TextField, 824  
textfield\_set\_textcolor  
    TextField, 825  
textfield\_set\_textmargins  
    TextField, 825  
textfield\_set\_underline  
    TextField, 826  
textfield\_set\_useellipsis  
    TextField, 826  
textfield\_set\_wantsreturn  
    TextField, 827  
textfield\_set\_wantstab  
    TextField, 827  
textfield\_set\_wordwrap  
    TextField, 827  
TextLayout, 828  
    JGRAPHICS\_TEXTLAYOUT\_NOWRAP, 829  
  
JGRAPHICS\_TEXTLAYOUT\_USEELLIPSIS, 829  
jtextlayout\_create, 830  
jtextlayout\_createpath, 830  
jtextlayout\_destroy, 830  
jtextlayout\_draw, 831  
jtextlayout\_getchar, 831  
jtextlayout\_getcharbox, 831  
jtextlayout\_getnumchars, 832  
jtextlayout\_measuretext, 832  
jtextlayout\_set, 833  
jtextlayout\_settext, 833  
jtextlayout\_settextcolor, 834  
jtextlayout\_withbgcolor, 834  
t\_jgraphics\_textlayout\_flags, 829  
  
Threads, 716  
    defer, 719  
    defer\_low, 720  
    e\_max\_systhread\_mutex\_flags, 719  
    isr, 721  
    schedule, 721  
    schedule\_delay, 722  
    systhread\_create, 723  
    systhread\_detach, 723  
    systhread\_equal, 724  
    systhread\_exit, 724  
    systhread\_getpriority, 725  
    systhread\_isaudiothread, 725  
    systhread\_ismainthread, 725  
    systhread\_istimerthread, 725  
    systhread\_join, 726  
    SYSTHREAD\_MUTEX\_ERRORCHECK, 719  
    SYSTHREAD\_MUTEX\_NORMAL, 719  
    SYSTHREAD\_MUTEX\_RECURSIVE, 719  
    systhread\_self, 726  
    systhread\_set\_name, 726  
    systhread\_setpriority, 727  
    systhread\_sleep, 727  
    systhread\_terminate, 727  
time\_calcquantize  
    ITM Time Objects, 709  
TIME\_FLAGS\_BBUSOURCE  
    ITM Time Objects, 700  
TIME\_FLAGS\_CHECKSCHEDULE  
    ITM Time Objects, 700  
TIME\_FLAGS\_EVENTLIST  
    ITM Time Objects, 700  
TIME\_FLAGS\_FIXED  
    ITM Time Objects, 700  
TIME\_FLAGS\_FIXEDONLY  
    ITM Time Objects, 700  
TIME\_FLAGS\_LISTENTICKS  
    ITM Time Objects, 700  
TIME\_FLAGS\_LOCATION  
    ITM Time Objects, 700

TIME\_FLAGS\_LOOKAHEAD  
     ITM Time Objects, 700  
 TIME\_FLAGS\_NOUNITS  
     ITM Time Objects, 700  
 TIME\_FLAGS\_PERMANENT  
     ITM Time Objects, 700  
 TIME\_FLAGS\_POSITIVE  
     ITM Time Objects, 700  
 TIME\_FLAGS\_TICKSONLY  
     ITM Time Objects, 700  
 TIME\_FLAGS\_TRANSPORT  
     ITM Time Objects, 700  
 TIME\_FLAGS\_USECLOCK  
     ITM Time Objects, 700  
 TIME\_FLAGS\_USEQELEM  
     ITM Time Objects, 700  
 time\_getitm  
     ITM Time Objects, 710  
 time\_getms  
     ITM Time Objects, 710  
 time\_getnamed  
     ITM Time Objects, 711  
 time\_getphase  
     ITM Time Objects, 711  
 time\_getticks  
     ITM Time Objects, 712  
 time\_isfixedunit  
     ITM Time Objects, 712  
 time\_listen  
     ITM Time Objects, 712  
 time\_new  
     ITM Time Objects, 713  
 time\_now  
     ITM Time Objects, 713  
 time\_schedule  
     ITM Time Objects, 714  
 time\_schedule\_limit  
     ITM Time Objects, 714  
 time\_setclock  
     ITM Time Objects, 714  
 time\_setvalue  
     ITM Time Objects, 715  
 time\_stop  
     ITM Time Objects, 715  
 time\_tick  
     ITM Time Objects, 715  
 Timing, 675  
 TWOPPI  
     MSP, 550  
 typedmess  
     Old-Style Classes, 267  
 Unicode, 847  
     charset\_convert, 848  
 charset\_unicodetoutf8, 849  
 charset\_utf8\_count, 849  
 charset\_utf8\_offset, 850  
 charset\_utf8tounicode, 850  
 User Interface, 736  
 vptr  
     MSP, 551  
 wind\_advise  
     Miscellaneous, 515  
 wind\_setcursor  
     Miscellaneous, 515  
 Windows, 541  
     jwind\_getactive, 541  
     jwind\_getat, 541  
     jwind\_getcount, 542  
 word, 1243  
 z\_dsp\_free  
     MSP, 555  
 z\_dsp\_setup  
     MSP, 555  
 zgetfn  
     Old-Style Classes, 268