

# Max5 API

## 5.0.8

Generated by Doxygen 1.6.1

Fri Sep 18 21:33:24 2009



# Contents

<b>1</b>	<b>Objects in C: A Roadmap</b>	<b>1</b>
1.1	Max Objects . . . . .	1
1.2	MSP Objects . . . . .	1
<b>2</b>	<b>Development System Information</b>	<b>3</b>
2.1	Building . . . . .	4
2.2	Mac . . . . .	4
2.2.1	XCode Project Setup . . . . .	4
2.2.2	Linking and Frameworks . . . . .	4
2.3	Windows . . . . .	5
2.3.1	Compiling with Cygwin . . . . .	5
2.3.1.1	Requirements . . . . .	5
2.3.1.2	Build Steps . . . . .	6
2.3.1.3	Additional Notes . . . . .	6
2.4	Important Project Settings . . . . .	7
2.4.1	Mac . . . . .	7
2.4.2	Windows . . . . .	7
2.5	Platform-specificity . . . . .	7
<b>3</b>	<b>Anatomy of a Max Object</b>	<b>9</b>
3.1	Include Files . . . . .	10
3.2	The Object Declaration . . . . .	10
3.3	Initialization Routine . . . . .	10
3.4	New Instance Routine . . . . .	11
3.5	Message Handlers . . . . .	12
<b>4</b>	<b>Inlets and Outlets</b>	<b>13</b>
4.1	Creating and Using Inlets . . . . .	14
4.2	Creating and Using Outlets . . . . .	15

4.3	Creating and Using Proxies . . . . .	16
4.4	Example . . . . .	16
<b>5</b>	<b>Atoms and Messages</b>	<b>19</b>
5.1	Argument Type Specifiers . . . . .	21
5.2	Writing A_GIMME Functions . . . . .	21
<b>6</b>	<b>The Scheduler</b>	<b>23</b>
6.1	Creating and Using Clocks . . . . .	24
6.2	Creating and Using Qelems . . . . .	25
6.3	Defer . . . . .	26
6.3.1	Defer Variants . . . . .	26
6.4	Schedule . . . . .	27
<b>7</b>	<b>Memory Allocation</b>	<b>29</b>
<b>8</b>	<b>Anatomy of a MSP Object</b>	<b>31</b>
<b>9</b>	<b>Advanced Signal Object Topics</b>	<b>35</b>
9.1	Saving Internal State . . . . .	36
9.2	Observing Patcher Muting . . . . .	36
9.3	Using Connection Information . . . . .	37
<b>10</b>	<b>Sending Messages, Calling Methods</b>	<b>39</b>
10.1	Attributes . . . . .	40
10.1.1	Attribute Basics . . . . .	40
10.1.2	Defining Attributes . . . . .	41
10.1.3	Attributes With Custom Getters and Setters . . . . .	41
10.2	Receiving Notifications . . . . .	42
<b>11</b>	<b>Anatomy of a UI Object</b>	<b>45</b>
11.1	Required Headers . . . . .	46
11.2	UI Object Data Structure . . . . .	47
11.3	Initialization Routine for UI Objects . . . . .	47
11.4	UI Object Methods . . . . .	48
11.5	Defining Attributes . . . . .	49
11.5.1	Standard Color Attribute . . . . .	51
11.5.2	Setting a Default Size . . . . .	51
11.6	New Instance Routine . . . . .	51

11.7 Dynamic Updating . . . . .	53
11.8 The Paint Method . . . . .	53
11.9 Handling Mouse Gestures . . . . .	55
11.10 Freeing a UI Object . . . . .	57
<b>12 File Handling</b>	<b>59</b>
12.1 Reading Text Files . . . . .	61
12.2 Reading Data Files . . . . .	61
12.3 Writing Files . . . . .	62
<b>13 Scripting the Patcher</b>	<b>65</b>
13.1 Knowing the Patcher . . . . .	66
13.1.1 Patcher Name and File Path . . . . .	66
13.1.2 Patcher Hierarchy . . . . .	66
13.1.3 Getting Objects in a Patcher . . . . .	67
13.1.4 Iteration Using Callbacks . . . . .	67
13.2 Creating Objects . . . . .	68
13.2.1 Connecting Objects . . . . .	68
13.3 Deleting Objects . . . . .	69
13.4 Obtaining and Changing Patcher and Object Attributes . . . . .	69
13.4.1 Patcher Attributes . . . . .	71
13.4.2 Box Attributes . . . . .	73
<b>14 Enhancements to Objects</b>	<b>75</b>
14.1 Preset Support . . . . .	76
14.2 Assistance . . . . .	76
14.3 Hot and Cold Inlets . . . . .	77
14.4 Showing a Text Editor . . . . .	77
14.5 Accessing Data in table Objects . . . . .	79
<b>15 Data Structures</b>	<b>81</b>
15.1 Linked Lists . . . . .	82
15.2 Hash Tables . . . . .	83
<b>16 Threading</b>	<b>85</b>
16.1 Max Threading Operation . . . . .	86
16.2 Thread Protection . . . . .	87
16.2.1 When Messages Arrive . . . . .	87
16.2.2 Critical Section Example . . . . .	87

<b>17 Drag'n'Drop</b>	<b>89</b>
17.1 Discussion . . . . .	90
<b>18 ITM</b>	<b>93</b>
18.1 Scheduling Temporary Events . . . . .	94
18.2 Permanent Events . . . . .	96
18.3 Cleaning Up . . . . .	97
<b>19 JGraphics</b>	<b>99</b>
<b>20 Appendix: Messages sent to Objects</b>	<b>101</b>
20.1 Messages for All Objects . . . . .	102
20.2 Messages for Non-UI Objects . . . . .	102
20.3 Messages for User Interface Objects . . . . .	103
20.4 Message for Audio Objects . . . . .	103
20.5 Messages for Objects Containing Text Fields . . . . .	104
20.6 Messages for Objects with Text Editor Windows . . . . .	104
20.7 Messages for Dataview Client Objects . . . . .	104
<b>21 Appendix: Providing Icons for UI Objects</b>	<b>105</b>
21.1 Object SVG Icon . . . . .	106
21.2 Object Palette Definition . . . . .	106
<b>22 Max Development for the Mac OS</b>	<b>109</b>
22.1 Development Environment . . . . .	110
22.2 Debugging . . . . .	110
22.3 What to check if it doesn't work... . . . .	110
<b>23 Max Development for Windows</b>	<b>111</b>
23.1 Development Environment . . . . .	112
23.2 Debugging . . . . .	112
23.3 What to check if it doesn't work... . . . .	112
<b>24 Module Index</b>	<b>113</b>
24.1 Modules . . . . .	113
<b>25 Data Structure Index</b>	<b>115</b>
25.1 Data Structures . . . . .	115
<b>26 Module Documentation</b>	<b>117</b>

26.1	Attributes	117
26.1.1	Detailed Description	130
26.1.2	Setting and Getting Attribute Values	131
26.1.2.1	Writing a custom Attribute Getter	131
26.1.2.2	Writing a custom Attribute Getter	131
26.1.3	Attribute Notificaton	131
26.1.4	Define Documentation	132
26.1.4.1	CLASS_ATTR_ACCESSORS	132
26.1.4.2	CLASS_ATTR_ADD_FLAGS	132
26.1.4.3	CLASS_ATTR_ALIAS	132
26.1.4.4	CLASS_ATTR_ATOM	133
26.1.4.5	CLASS_ATTR_ATOM_ARRAY	133
26.1.4.6	CLASS_ATTR_ATOM_VARSIZE	133
26.1.4.7	CLASS_ATTR_CATEGORY	134
26.1.4.8	CLASS_ATTR_CHAR	134
26.1.4.9	CLASS_ATTR_CHAR_ARRAY	135
26.1.4.10	CLASS_ATTR_CHAR_VARSIZE	135
26.1.4.11	CLASS_ATTR_DEFAULT	135
26.1.4.12	CLASS_ATTR_DEFAULT_PAINT	136
26.1.4.13	CLASS_ATTR_DEFAULT_SAVE	136
26.1.4.14	CLASS_ATTR_DEFAULT_SAVE_PAINT	136
26.1.4.15	CLASS_ATTR_DEFAULTNAME	137
26.1.4.16	CLASS_ATTR_DEFAULTNAME_PAINT	137
26.1.4.17	CLASS_ATTR_DEFAULTNAME_SAVE	138
26.1.4.18	CLASS_ATTR_DEFAULTNAME_SAVE_PAINT	138
26.1.4.19	CLASS_ATTR_DOUBLE	138
26.1.4.20	CLASS_ATTR_DOUBLE_ARRAY	139
26.1.4.21	CLASS_ATTR_DOUBLE_VARSIZE	139
26.1.4.22	CLASS_ATTR_ENUM	140
26.1.4.23	CLASS_ATTR_ENUMINDEX	140
26.1.4.24	CLASS_ATTR_FILTER_CLIP	141
26.1.4.25	CLASS_ATTR_FILTER_MAX	141
26.1.4.26	CLASS_ATTR_FILTER_MIN	141
26.1.4.27	CLASS_ATTR_FLOAT	142
26.1.4.28	CLASS_ATTR_FLOAT_ARRAY	142
26.1.4.29	CLASS_ATTR_FLOAT_VARSIZE	143

26.1.4.30 CLASS_ATTR_INVISIBLE . . . . .	143
26.1.4.31 CLASS_ATTR_LABEL . . . . .	143
26.1.4.32 CLASS_ATTR_LONG . . . . .	144
26.1.4.33 CLASS_ATTR_LONG_ARRAY . . . . .	144
26.1.4.34 CLASS_ATTR_LONG_VARSIZE . . . . .	144
26.1.4.35 CLASS_ATTR_MAX . . . . .	145
26.1.4.36 CLASS_ATTR_MIN . . . . .	145
26.1.4.37 CLASS_ATTR_OBJ . . . . .	146
26.1.4.38 CLASS_ATTR_OBJ_ARRAY . . . . .	146
26.1.4.39 CLASS_ATTR_OBJ_VARSIZE . . . . .	146
26.1.4.40 CLASS_ATTR_ORDER . . . . .	147
26.1.4.41 CLASS_ATTR_PAINT . . . . .	147
26.1.4.42 CLASS_ATTR_REMOVE_FLAGS . . . . .	147
26.1.4.43 CLASS_ATTR_RGBA . . . . .	148
26.1.4.44 CLASS_ATTR_SAVE . . . . .	148
26.1.4.45 CLASS_ATTR_STYLE . . . . .	148
26.1.4.46 CLASS_ATTR_STYLE_LABEL . . . . .	149
26.1.4.47 CLASS_ATTR_SYM . . . . .	149
26.1.4.48 CLASS_ATTR_SYM_ARRAY . . . . .	150
26.1.4.49 CLASS_ATTR_SYM_VARSIZE . . . . .	150
26.1.4.50 CLASS_STICKY_ATTR . . . . .	151
26.1.4.51 CLASS_STICKY_ATTR_CLEAR . . . . .	151
26.1.4.52 CLASS_STICKY_METHOD . . . . .	152
26.1.4.53 CLASS_STICKY_METHOD_CLEAR . . . . .	152
26.1.4.54 OBJ_ATTR_ATOM . . . . .	152
26.1.4.55 OBJ_ATTR_ATOM_ARRAY . . . . .	153
26.1.4.56 OBJ_ATTR_CHAR . . . . .	153
26.1.4.57 OBJ_ATTR_CHAR_ARRAY . . . . .	153
26.1.4.58 OBJ_ATTR_DEFAULT . . . . .	154
26.1.4.59 OBJ_ATTR_DEFAULT_SAVE . . . . .	154
26.1.4.60 OBJ_ATTR_DOUBLE . . . . .	154
26.1.4.61 OBJ_ATTR_DOUBLE_ARRAY . . . . .	155
26.1.4.62 OBJ_ATTR_FLOAT . . . . .	155
26.1.4.63 OBJ_ATTR_FLOAT_ARRAY . . . . .	155
26.1.4.64 OBJ_ATTR_LONG . . . . .	155
26.1.4.65 OBJ_ATTR_LONG_ARRAY . . . . .	156



26.1.4.66 OBJ_ATTR_OBJ	156
26.1.4.67 OBJ_ATTR_OBJ_ARRAY	156
26.1.4.68 OBJ_ATTR_SAVE	157
26.1.4.69 OBJ_ATTR_SYM	157
26.1.4.70 OBJ_ATTR_SYM_ARRAY	157
26.1.4.71 STATIC_ATTR_ATOM	157
26.1.4.72 STATIC_ATTR_ATOM_ARRAY	158
26.1.4.73 STATIC_ATTR_CHAR	158
26.1.4.74 STATIC_ATTR_CHAR_ARRAY	158
26.1.4.75 STATIC_ATTR_DOUBLE	159
26.1.4.76 STATIC_ATTR_DOUBLE_ARRAY	159
26.1.4.77 STATIC_ATTR_FLOAT	159
26.1.4.78 STATIC_ATTR_FLOAT_ARRAY	159
26.1.4.79 STATIC_ATTR_LONG	160
26.1.4.80 STATIC_ATTR_LONG_ARRAY	160
26.1.4.81 STATIC_ATTR_OBJ	160
26.1.4.82 STATIC_ATTR_OBJ_ARRAY	161
26.1.4.83 STATIC_ATTR_SYM	161
26.1.4.84 STATIC_ATTR_SYM_ARRAY	161
26.1.4.85 STRUCT_ATTR_ATOM	161
26.1.4.86 STRUCT_ATTR_ATOM_ARRAY	162
26.1.4.87 STRUCT_ATTR_ATOM_VARSIZE	162
26.1.4.88 STRUCT_ATTR_CHAR	163
26.1.4.89 STRUCT_ATTR_CHAR_ARRAY	163
26.1.4.90 STRUCT_ATTR_CHAR_VARSIZE	163
26.1.4.91 STRUCT_ATTR_DOUBLE	164
26.1.4.92 STRUCT_ATTR_DOUBLE_ARRAY	164
26.1.4.93 STRUCT_ATTR_DOUBLE_VARSIZE	164
26.1.4.94 STRUCT_ATTR_FLOAT	165
26.1.4.95 STRUCT_ATTR_FLOAT_ARRAY	165
26.1.4.96 STRUCT_ATTR_FLOAT_VARSIZE	165
26.1.4.97 STRUCT_ATTR_LONG	166
26.1.4.98 STRUCT_ATTR_LONG_ARRAY	166
26.1.4.99 STRUCT_ATTR_LONG_VARSIZE	166
26.1.4.100 STRUCT_ATTR_OBJ	167
26.1.4.101 STRUCT_ATTR_OBJ_ARRAY	167

26.1.4.102	STRUCT_ATTR_OBJ_VARSIZE	167
26.1.4.103	STRUCT_ATTR_SYM	168
26.1.4.104	STRUCT_ATTR_SYM_ARRAY	168
26.1.4.105	STRUCT_ATTR_SYM_VARSIZE	168
26.1.5	Enumeration Type Documentation	169
26.1.5.1	e_max_attrflags	169
26.1.6	Function Documentation	169
26.1.6.1	attr_addfilter_clip	169
26.1.6.2	attr_addfilter_clip_scale	170
26.1.6.3	attr_addfilterget_clip	170
26.1.6.4	attr_addfilterget_clip_scale	171
26.1.6.5	attr_addfilterget_proc	171
26.1.6.6	attr_addfilterset_clip	172
26.1.6.7	attr_addfilterset_clip_scale	172
26.1.6.8	attr_addfilterset_proc	172
26.1.6.9	attr_args_dictionary	173
26.1.6.10	attr_args_offset	173
26.1.6.11	attr_args_process	174
26.1.6.12	attr_dictionary_process	174
26.1.6.13	attr_offset_array_new	175
26.1.6.14	attr_offset_new	175
26.1.6.15	attribute_new	176
26.1.6.16	object_addattr	177
26.1.6.17	object_attr_get	177
26.1.6.18	object_attr_get_rect	178
26.1.6.19	object_attr_getchar_array	178
26.1.6.20	object_attr_getcolor	179
26.1.6.21	object_attr_getdouble_array	179
26.1.6.22	object_attr_getdump	179
26.1.6.23	object_attr_getfloat	180
26.1.6.24	object_attr_getfloat_array	180
26.1.6.25	object_attr_getjrgba	180
26.1.6.26	object_attr_getlong	181
26.1.6.27	object_attr_getlong_array	181
26.1.6.28	object_attr_getpt	181
26.1.6.29	object_attr_getsize	182

26.1.6.30	<a href="#">object_attr_getsym</a>	182
26.1.6.31	<a href="#">object_attr_getsym_array</a>	182
26.1.6.32	<a href="#">object_attr_method</a>	183
26.1.6.33	<a href="#">object_attr_set_rect</a>	183
26.1.6.34	<a href="#">object_attr_setchar_array</a>	183
26.1.6.35	<a href="#">object_attr_setcolor</a>	184
26.1.6.36	<a href="#">object_attr_setdouble_array</a>	184
26.1.6.37	<a href="#">object_attr_setfloat</a>	184
26.1.6.38	<a href="#">object_attr_setfloat_array</a>	185
26.1.6.39	<a href="#">object_attr_setjrgba</a>	185
26.1.6.40	<a href="#">object_attr_setlong</a>	185
26.1.6.41	<a href="#">object_attr_setlong_array</a>	186
26.1.6.42	<a href="#">object_attr_setparse</a>	186
26.1.6.43	<a href="#">object_attr_setpt</a>	186
26.1.6.44	<a href="#">object_attr_setsize</a>	187
26.1.6.45	<a href="#">object_attr_setsym</a>	187
26.1.6.46	<a href="#">object_attr_setsym_array</a>	187
26.1.6.47	<a href="#">object_attr_setvalueof</a>	188
26.1.6.48	<a href="#">object_attr_usercanget</a>	188
26.1.6.49	<a href="#">object_attr_usercanset</a>	188
26.1.6.50	<a href="#">object_chuckattr</a>	188
26.1.6.51	<a href="#">object_deleteattr</a>	189
26.1.6.52	<a href="#">object_new_parse</a>	189
26.2	<a href="#">Classes</a>	190
26.2.1	<a href="#">Detailed Description</a>	192
26.2.2	<a href="#">Define Documentation</a>	192
26.2.2.1	<a href="#">CLASS_BOX</a>	192
26.2.3	<a href="#">Enumeration Type Documentation</a>	192
26.2.3.1	<a href="#">e_max_class_flags</a>	192
26.2.4	<a href="#">Function Documentation</a>	193
26.2.4.1	<a href="#">class_addattr</a>	193
26.2.4.2	<a href="#">class_addmethod</a>	193
26.2.4.3	<a href="#">class_alias</a>	193
26.2.4.4	<a href="#">class_dumpout_wrap</a>	194
26.2.4.5	<a href="#">class_findbyname</a>	194
26.2.4.6	<a href="#">class_findbyname_casefree</a>	194

26.2.4.7	<a href="#">class_free</a>	194
26.2.4.8	<a href="#">class_is_ui</a>	195
26.2.4.9	<a href="#">class_nameget</a>	195
26.2.4.10	<a href="#">class_new</a>	195
26.2.4.11	<a href="#">class_obexoffset_get</a>	196
26.2.4.12	<a href="#">class_obexoffset_set</a>	196
26.2.4.13	<a href="#">class_register</a>	196
26.3	Old-Style Classes	197
26.3.1	Function Documentation	198
26.3.1.1	<a href="#">addbang</a>	198
26.3.1.2	<a href="#">addfloat</a>	198
26.3.1.3	<a href="#">addftx</a>	198
26.3.1.4	<a href="#">addint</a>	198
26.3.1.5	<a href="#">addinx</a>	199
26.3.1.6	<a href="#">addmess</a>	199
26.3.1.7	<a href="#">alias</a>	199
26.3.1.8	<a href="#">class_setname</a>	199
26.3.1.9	<a href="#">egetfn</a>	200
26.3.1.10	<a href="#">freeobject</a>	200
26.3.1.11	<a href="#">getfn</a>	200
26.3.1.12	<a href="#">newinstance</a>	201
26.3.1.13	<a href="#">newobject</a>	201
26.3.1.14	<a href="#">setup</a>	201
26.3.1.15	<a href="#">typedmess</a>	202
26.3.1.16	<a href="#">zgetfn</a>	202
26.4	Inlets and Outlets	203
26.4.1	Detailed Description	204
26.4.2	Function Documentation	204
26.4.2.1	<a href="#">bangout</a>	204
26.4.2.2	<a href="#">floatin</a>	204
26.4.2.3	<a href="#">floatout</a>	204
26.4.2.4	<a href="#">inlet_new</a>	205
26.4.2.5	<a href="#">intin</a>	205
26.4.2.6	<a href="#">intout</a>	206
26.4.2.7	<a href="#">listout</a>	206
26.4.2.8	<a href="#">outlet_anything</a>	206

26.4.2.9	outlet_bang	207
26.4.2.10	outlet_float	207
26.4.2.11	outlet_int	207
26.4.2.12	outlet_list	208
26.4.2.13	outlet_new	208
26.4.2.14	proxy_getinlet	209
26.4.2.15	proxy_new	209
26.5	Data Storage	210
26.5.1	Detailed Description	211
26.5.2	Typedef Documentation	211
26.5.2.1	t_cmpfn	211
26.5.3	Enumeration Type Documentation	212
26.5.3.1	e_max_datastore_flags	212
26.6	Atom Array	213
26.6.1	Detailed Description	214
26.6.2	Function Documentation	214
26.6.2.1	atomarray_appendatom	214
26.6.2.2	atomarray_appendatoms	214
26.6.2.3	atomarray_chuckindex	215
26.6.2.4	atomarray_clear	215
26.6.2.5	atomarray_copyatoms	215
26.6.2.6	atomarray_duplicate	216
26.6.2.7	atomarray_funall	216
26.6.2.8	atomarray_getatoms	216
26.6.2.9	atomarray_getindex	217
26.6.2.10	atomarray_getsize	217
26.6.2.11	atomarray_new	217
26.6.2.12	atomarray_setatoms	218
26.7	Database	219
26.7.1	Detailed Description	221
26.7.2	Typedef Documentation	221
26.7.2.1	t_database	221
26.7.2.2	t_db_result	221
26.7.2.3	t_db_view	221
26.7.3	Function Documentation	221
26.7.3.1	db_close	221

26.7.3.2	<code>db_open</code>	222
26.7.3.3	<code>db_query</code>	223
26.7.3.4	<code>db_query_getlastinsertid</code>	223
26.7.3.5	<code>db_query_silent</code>	224
26.7.3.6	<code>db_query_table_addcolumn</code>	224
26.7.3.7	<code>db_query_table_new</code>	225
26.7.3.8	<code>db_result_clear</code>	225
26.7.3.9	<code>db_result_datetimeinseconds</code>	226
26.7.3.10	<code>db_result_fieldname</code>	226
26.7.3.11	<code>db_result_float</code>	226
26.7.3.12	<code>db_result_long</code>	227
26.7.3.13	<code>db_result_nextrecord</code>	227
26.7.3.14	<code>db_result_numfields</code>	228
26.7.3.15	<code>db_result_numrecords</code>	228
26.7.3.16	<code>db_result_reset</code>	229
26.7.3.17	<code>db_result_string</code>	229
26.7.3.18	<code>db_transaction_end</code>	229
26.7.3.19	<code>db_transaction_flush</code>	230
26.7.3.20	<code>db_transaction_start</code>	230
26.7.3.21	<code>db_util_datetostring</code>	231
26.7.3.22	<code>db_util_stringtodate</code>	231
26.7.3.23	<code>db_view_create</code>	231
26.7.3.24	<code>db_view_getresult</code>	232
26.7.3.25	<code>db_view_remove</code>	232
26.7.3.26	<code>db_view_setquery</code>	233
26.8	Dictionary	234
26.8.1	Detailed Description	237
26.8.2	Using Dictionaries	237
26.8.2.1	Understanding Dictionaries	237
26.8.2.2	When to Free a Dictionary	238
26.8.2.3	Some Common Uses of Dictionaries	239
26.8.3	Function Documentation	239
26.8.3.1	<code>dictionary_appendatom</code>	239
26.8.3.2	<code>dictionary_appendatomarray</code>	239
26.8.3.3	<code>dictionary_appendatoms</code>	240
26.8.3.4	<code>dictionary_appenddictionary</code>	240

26.8.3.5	<a href="#">dictionary_appendfloat</a>	241
26.8.3.6	<a href="#">dictionary_appendlong</a>	241
26.8.3.7	<a href="#">dictionary_appendobject</a>	241
26.8.3.8	<a href="#">dictionary_appendstring</a>	242
26.8.3.9	<a href="#">dictionary_appendsym</a>	242
26.8.3.10	<a href="#">dictionary_chuckentry</a>	242
26.8.3.11	<a href="#">dictionary_clear</a>	243
26.8.3.12	<a href="#">dictionary_copyatoms</a>	243
26.8.3.13	<a href="#">dictionary_copydefatoms</a>	243
26.8.3.14	<a href="#">dictionary_copyentries</a>	244
26.8.3.15	<a href="#">dictionary_copyunique</a>	244
26.8.3.16	<a href="#">dictionary_deleteentry</a>	245
26.8.3.17	<a href="#">dictionary_dump</a>	245
26.8.3.18	<a href="#">dictionary_entry_getkey</a>	245
26.8.3.19	<a href="#">dictionary_entry_getvalue</a>	246
26.8.3.20	<a href="#">dictionary_entryisatomarray</a>	246
26.8.3.21	<a href="#">dictionary_entryisdictionary</a>	246
26.8.3.22	<a href="#">dictionary_entryisstring</a>	246
26.8.3.23	<a href="#">dictionary_freekeys</a>	247
26.8.3.24	<a href="#">dictionary_funall</a>	247
26.8.3.25	<a href="#">dictionary_getatom</a>	247
26.8.3.26	<a href="#">dictionary_getatomarray</a>	248
26.8.3.27	<a href="#">dictionary_getatoms</a>	248
26.8.3.28	<a href="#">dictionary_getdefatom</a>	248
26.8.3.29	<a href="#">dictionary_getdefatoms</a>	249
26.8.3.30	<a href="#">dictionary_getdeffloat</a>	249
26.8.3.31	<a href="#">dictionary_getdeflong</a>	250
26.8.3.32	<a href="#">dictionary_getdefstring</a>	250
26.8.3.33	<a href="#">dictionary_getdefsym</a>	250
26.8.3.34	<a href="#">dictionary_getdictionary</a>	251
26.8.3.35	<a href="#">dictionary_getentrycount</a>	251
26.8.3.36	<a href="#">dictionary_getfloat</a>	251
26.8.3.37	<a href="#">dictionary_getkeys</a>	252
26.8.3.38	<a href="#">dictionary_getlong</a>	252
26.8.3.39	<a href="#">dictionary_getobject</a>	252
26.8.3.40	<a href="#">dictionary_getstring</a>	253

26.8.3.41	<a href="#">dictionary_getsym</a>	253
26.8.3.42	<a href="#">dictionary_hasentry</a>	253
26.8.3.43	<a href="#">dictionary_new</a>	254
26.8.3.44	<a href="#">dictionary_read</a>	254
26.8.3.45	<a href="#">dictionary_sprintf</a>	254
26.8.3.46	<a href="#">dictionary_write</a>	255
26.8.3.47	<a href="#">postdictionary</a>	255
26.9	<a href="#">Hash Table</a>	256
26.9.1	<a href="#">Detailed Description</a>	257
26.9.2	<a href="#">Define Documentation</a>	258
26.9.2.1	<a href="#">HASH_DEFSLOTS</a>	258
26.9.3	<a href="#">Function Documentation</a>	258
26.9.3.1	<a href="#">hashtab_chuck</a>	258
26.9.3.2	<a href="#">hashtab_chuckkey</a>	258
26.9.3.3	<a href="#">hashtab_clear</a>	259
26.9.3.4	<a href="#">hashtab_delete</a>	259
26.9.3.5	<a href="#">hashtab_findfirst</a>	259
26.9.3.6	<a href="#">hashtab_flags</a>	260
26.9.3.7	<a href="#">hashtab_funall</a>	260
26.9.3.8	<a href="#">hashtab_getflags</a>	260
26.9.3.9	<a href="#">hashtab_getkeyflags</a>	261
26.9.3.10	<a href="#">hashtab_getkeys</a>	261
26.9.3.11	<a href="#">hashtab_getsize</a>	261
26.9.3.12	<a href="#">hashtab_keyflags</a>	262
26.9.3.13	<a href="#">hashtab_lookup</a>	262
26.9.3.14	<a href="#">hashtab_lookupflags</a>	262
26.9.3.15	<a href="#">hashtab_methodall</a>	263
26.9.3.16	<a href="#">hashtab_new</a>	263
26.9.3.17	<a href="#">hashtab_print</a>	264
26.9.3.18	<a href="#">hashtab_readonly</a>	264
26.9.3.19	<a href="#">hashtab_store</a>	264
26.9.3.20	<a href="#">hashtab_store_safe</a>	264
26.9.3.21	<a href="#">hashtab_storeflags</a>	265
26.10	<a href="#">Index Map</a>	266
26.10.1	<a href="#">Detailed Description</a>	267
26.10.2	<a href="#">Function Documentation</a>	267



26.10.2.1	indexmap_append	267
26.10.2.2	indexmap_clear	267
26.10.2.3	indexmap_datafromindex	267
26.10.2.4	indexmap_delete	268
26.10.2.5	indexmap_delete_index	268
26.10.2.6	indexmap_delete_index_multi	268
26.10.2.7	indexmap_delete_multi	268
26.10.2.8	indexmap_getsize	269
26.10.2.9	indexmap_indexfromdata	269
26.10.2.10	indexmap_move	269
26.10.2.11	indexmap_new	269
26.10.2.12	indexmap_sort	270
26.11	Linked List	271
26.11.1	Detailed Description	273
26.11.2	Function Documentation	273
26.11.2.1	linklist_append	273
26.11.2.2	linklist_chuck	274
26.11.2.3	linklist_chuckindex	274
26.11.2.4	linklist_chuckobject	274
26.11.2.5	linklist_clear	275
26.11.2.6	linklist_deleteindex	275
26.11.2.7	linklist_findall	275
26.11.2.8	linklist_findfirst	276
26.11.2.9	linklist_flags	277
26.11.2.10	linklist_funall	277
26.11.2.11	linklist_funall_break	277
26.11.2.12	linklist_funindex	278
26.11.2.13	linklist_getflags	278
26.11.2.14	linklist_getindex	278
26.11.2.15	linklist_getsize	279
26.11.2.16	linklist_insert_sorted	279
26.11.2.17	linklist_insertafterobjptr	279
26.11.2.18	linklist_insertbeforeobjptr	279
26.11.2.19	linklist_insertindex	280
26.11.2.20	linklist_last	280
26.11.2.21	linklist_makearray	280

26.11.2.22	linklist_match	280
26.11.2.23	linklist_methodall	281
26.11.2.24	linklist_methodindex	281
26.11.2.25	linklist_moveafterobjptr	281
26.11.2.26	linklist_movebeforeobjptr	282
26.11.2.27	linklist_new	282
26.11.2.28	linklist_next	282
26.11.2.29	linklist_objptr2index	282
26.11.2.30	linklist_prev	283
26.11.2.31	linklist_readonly	283
26.11.2.32	linklist_reverse	283
26.11.2.33	linklist_rotate	283
26.11.2.34	linklist_shuffle	284
26.11.2.35	linklist_sort	284
26.11.2.36	linklist_substitute	284
26.11.2.37	linklist_swap	285
26.12	Quick Map	286
26.12.1	Detailed Description	286
26.12.2	Function Documentation	286
26.12.2.1	quickmap_add	286
26.12.2.2	quickmap_drop	287
26.12.2.3	quickmap_lookup_key1	287
26.12.2.4	quickmap_lookup_key2	287
26.12.2.5	quickmap_new	288
26.12.2.6	quickmap_readonly	288
26.13	String Object	289
26.13.1	Detailed Description	289
26.13.2	Function Documentation	289
26.13.2.1	string_getptr	289
26.13.2.2	string_new	290
26.14	Symbol Object	291
26.14.1	Detailed Description	291
26.14.2	Function Documentation	291
26.14.2.1	symobject_linklist_match	291
26.14.2.2	symobject_new	292
26.15	Data Types	293

26.15.1 Typedef Documentation . . . . .	294
26.15.1.1 t_max_err . . . . .	294
26.16 Atoms . . . . .	295
26.16.1 Enumeration Type Documentation . . . . .	298
26.16.1.1 e_max_atom_gettext_flags . . . . .	298
26.16.1.2 e_max_atomtypes . . . . .	299
26.16.2 Function Documentation . . . . .	300
26.16.2.1 atom_alloc . . . . .	300
26.16.2.2 atom_alloc_array . . . . .	300
26.16.2.3 atom_arg_getdouble . . . . .	300
26.16.2.4 atom_arg_getfloat . . . . .	301
26.16.2.5 atom_arg_getlong . . . . .	301
26.16.2.6 atom_arg_getobjclass . . . . .	302
26.16.2.7 atom_arg_getsym . . . . .	302
26.16.2.8 atom_copy . . . . .	303
26.16.2.9 atom_getatom_array . . . . .	303
26.16.2.10 atom_getchar_array . . . . .	303
26.16.2.11 atom_getcharfix . . . . .	304
26.16.2.12 atom_getdouble_array . . . . .	304
26.16.2.13 atom_getfloat . . . . .	304
26.16.2.14 atom_getfloat_array . . . . .	305
26.16.2.15 atom_getformat . . . . .	305
26.16.2.16 atom_getlong . . . . .	306
26.16.2.17 atom_getlong_array . . . . .	306
26.16.2.18 atom_getobj . . . . .	306
26.16.2.19 atom_getobj_array . . . . .	306
26.16.2.20 atom_getobjclass . . . . .	307
26.16.2.21 atom_getsym . . . . .	307
26.16.2.22 atom_getsym_array . . . . .	307
26.16.2.23 atom_gettext . . . . .	308
26.16.2.24 atom_gettype . . . . .	308
26.16.2.25 atom_setatom_array . . . . .	308
26.16.2.26 atom_setchar_array . . . . .	309
26.16.2.27 atom_setdouble_array . . . . .	309
26.16.2.28 atom_setfloat . . . . .	309
26.16.2.29 atom_setfloat_array . . . . .	310

26.16.2.30	atom_setformat	310
26.16.2.31	atom_setlong	310
26.16.2.32	atom_setlong_array	311
26.16.2.33	atom_setobj	311
26.16.2.34	atom_setobj_array	311
26.16.2.35	atom_setparse	312
26.16.2.36	atom_setsym	312
26.16.2.37	atom_setsym_array	312
26.16.2.38	atom_is_atomarray	313
26.16.2.39	atom_is_dictionary	313
26.16.2.40	atom_is_string	313
26.16.2.41	postargs	313
26.17	Atombufs	314
26.17.1	Detailed Description	314
26.17.2	Function Documentation	314
26.17.2.1	atombuf_free	314
26.17.2.2	atombuf_new	314
26.17.2.3	atombuf_text	315
26.18	Binbufs	316
26.18.1	Detailed Description	316
26.18.2	Function Documentation	317
26.18.2.1	binbuf_append	317
26.18.2.2	binbuf_eval	317
26.18.2.3	binbuf_getatom	317
26.18.2.4	binbuf_insert	318
26.18.2.5	binbuf_new	318
26.18.2.6	binbuf_set	318
26.18.2.7	binbuf_text	318
26.18.2.8	binbuf_totext	319
26.18.2.9	binbuf_vinsert	319
26.18.2.10	readatom	320
26.19	Symbols	322
26.19.1	Detailed Description	322
26.19.2	Function Documentation	323
26.19.2.1	gensym	323
26.20	Files and Folders	324

26.20.1 Detailed Description . . . . .	328
26.20.2 The Sysfile API . . . . .	329
26.20.3 Example: filebyte (notes from the IRCAM workshop) . . . . .	329
26.20.3.1 Paths . . . . .	329
26.20.3.2 t_filehandle . . . . .	329
26.20.3.3 File Names . . . . .	329
26.20.3.4 File Path Names . . . . .	329
26.20.4 Collectives and Fileusage . . . . .	330
26.20.5 Filewatchers . . . . .	330
26.20.6 Define Documentation . . . . .	330
26.20.6.1 MAX_FILENAME_CHARS . . . . .	330
26.20.7 Typedef Documentation . . . . .	330
26.20.7.1 t_filehandle . . . . .	330
26.20.8 Enumeration Type Documentation . . . . .	330
26.20.8.1 e_max_fileinfo_flags . . . . .	330
26.20.8.2 e_max_openfile_permissions . . . . .	331
26.20.8.3 e_max_path_folder_flags . . . . .	331
26.20.8.4 e_max_path_styles . . . . .	331
26.20.8.5 e_max_path_types . . . . .	331
26.20.8.6 e_max_sysfile_posmodes . . . . .	332
26.20.8.7 e_max_sysfile_textflags . . . . .	332
26.20.9 Function Documentation . . . . .	332
26.20.9.1 fileusage_addfile . . . . .	332
26.20.9.2 filewatcher_new . . . . .	333
26.20.9.3 locatefile . . . . .	333
26.20.9.4 locatefile_extended . . . . .	334
26.20.9.5 locatefiletype . . . . .	334
26.20.9.6 open_dialog . . . . .	335
26.20.9.7 open_promptset . . . . .	335
26.20.9.8 path_closefolder . . . . .	336
26.20.9.9 path_createsysfile . . . . .	336
26.20.9.10path_fileinfo . . . . .	336
26.20.9.11path_foldernextfile . . . . .	336
26.20.9.12path_frompathname . . . . .	337
26.20.9.13path_getapppath . . . . .	337
26.20.9.14path_getdefault . . . . .	337

26.20.9.15	path_getfilemoddate . . . . .	337
26.20.9.16	path_getmoddate . . . . .	338
26.20.9.17	path_nameconform . . . . .	338
26.20.9.18	path_openfolder . . . . .	338
26.20.9.19	path_opensysfile . . . . .	339
26.20.9.20	path_resolvefile . . . . .	339
26.20.9.21	path_setdefault . . . . .	339
26.20.9.22	path_topathname . . . . .	339
26.20.9.23	path_topotentialname . . . . .	340
26.20.9.24	saveas_dialog . . . . .	340
26.20.9.25	saveas_promptset . . . . .	341
26.20.9.26	saveasdialog_extended . . . . .	341
26.20.9.27	sysfile_close . . . . .	342
26.20.9.28	sysfile_geteof . . . . .	342
26.20.9.29	sysfile_getpos . . . . .	343
26.20.9.30	sysfile_openhandle . . . . .	343
26.20.9.31	sysfile_openptrsize . . . . .	343
26.20.9.32	sysfile_read . . . . .	343
26.20.9.33	sysfile_readtextfile . . . . .	344
26.20.9.34	sysfile_readtohandle . . . . .	344
26.20.9.35	sysfile_readtoptr . . . . .	344
26.20.9.36	sysfile_seteof . . . . .	345
26.20.9.37	sysfile_setpos . . . . .	345
26.20.9.38	sysfile_spoolcopy . . . . .	345
26.20.9.39	sysfile_write . . . . .	346
26.20.9.40	sysfile_writetextfile . . . . .	346
26.21	Memory Management . . . . .	347
26.21.1	Detailed Description . . . . .	348
26.21.2	Sysmem API . . . . .	349
26.21.3	Define Documentation . . . . .	349
26.21.3.1	MM_UNIFIED . . . . .	349
26.21.4	Function Documentation . . . . .	349
26.21.4.1	disposhandle . . . . .	349
26.21.4.2	freebytes . . . . .	349
26.21.4.3	freebytes16 . . . . .	349
26.21.4.4	getbytes . . . . .	350

26.21.4.5	getbytes16	350
26.21.4.6	growhandle	350
26.21.4.7	newhandle	351
26.21.4.8	sysmem_copyptr	351
26.21.4.9	sysmem_freehandle	351
26.21.4.10	sysmem_freeptr	351
26.21.4.11	sysmem_handlesize	351
26.21.4.12	sysmem_lockhandle	352
26.21.4.13	sysmem_newhandle	352
26.21.4.14	sysmem_newhandleclear	352
26.21.4.15	sysmem_newptr	352
26.21.4.16	sysmem_newptrclear	353
26.21.4.17	sysmem_nullterminatehandle	353
26.21.4.18	sysmem_ptrandhand	353
26.21.4.19	sysmem_ptrbeforehand	353
26.21.4.20	sysmem_ptrsize	354
26.21.4.21	sysmem_resizehandle	354
26.21.4.22	sysmem_resizeptr	354
26.21.4.23	sysmem_resizeptrclear	354
26.22	Miscellaneous	355
26.22.1	Define Documentation	358
26.22.1.1	BEGIN_USING_C_LINKAGE	358
26.22.1.2	calcoffset	358
26.22.1.3	CLIP	358
26.22.1.4	InRange	359
26.22.1.5	MAX	359
26.22.1.6	MIN	359
26.22.2	Enumeration Type Documentation	360
26.22.2.1	e_max_errorcodes	360
26.22.2.2	e_max_wind_advise_result	360
26.22.3	Function Documentation	360
26.22.3.1	error_subscribe	360
26.22.3.2	error_sym	361
26.22.3.3	error_unsubscribe	361
26.22.3.4	globalsymbol_bind	361
26.22.3.5	globalsymbol_dereference	361

26.22.3.6	<a href="#">globalsymbol_reference</a>	362
26.22.3.7	<a href="#">globalsymbol_unbind</a>	362
26.22.3.8	<a href="#">maxversion</a>	362
26.22.3.9	<a href="#">object_obex_quickref</a>	363
26.22.3.10	<a href="#">post_sym</a>	363
26.22.3.11	<a href="#">quittask_install</a>	363
26.22.3.12	<a href="#">quittask_remove</a>	363
26.22.3.13	<a href="#">snprintf_zero</a>	364
26.22.3.14	<a href="#">strncat_zero</a>	364
26.22.3.15	<a href="#">strncpy_zero</a>	364
26.22.3.16	<a href="#">symbol_unique</a>	364
26.22.3.17	<a href="#">symbolarray_sort</a>	365
26.22.3.18	<a href="#">wind_advise</a>	365
26.22.3.19	<a href="#">wind_setcursor</a>	365
26.23	<a href="#">Console</a>	366
26.23.1	<a href="#">Function Documentation</a>	366
26.23.1.1	<a href="#">cpost</a>	366
26.23.1.2	<a href="#">error</a>	367
26.23.1.3	<a href="#">object_error</a>	367
26.23.1.4	<a href="#">object_error_obtrusive</a>	368
26.23.1.5	<a href="#">object_post</a>	368
26.23.1.6	<a href="#">object_warn</a>	369
26.23.1.7	<a href="#">ouchstring</a>	369
26.23.1.8	<a href="#">post</a>	369
26.23.1.9	<a href="#">postatom</a>	370
26.24	<a href="#">Byte Ordering</a>	371
26.24.1	<a href="#">Detailed Description</a>	372
26.24.2	<a href="#">Define Documentation</a>	372
26.24.2.1	<a href="#">BYTEORDER_LSBF32</a>	372
26.24.2.2	<a href="#">BYTEORDER_LSBF64</a>	372
26.24.2.3	<a href="#">BYTEORDER_LSBW16</a>	373
26.24.2.4	<a href="#">BYTEORDER_LSBW32</a>	373
26.24.2.5	<a href="#">BYTEORDER_MSBF32</a>	373
26.24.2.6	<a href="#">BYTEORDER_MSBF64</a>	373
26.24.2.7	<a href="#">BYTEORDER_MSBW16</a>	374
26.24.2.8	<a href="#">BYTEORDER_MSBW32</a>	374



26.24.2.9	BYTEORDER_SWAPF32	374
26.24.2.10	BYTEORDER_SWAPF64	375
26.24.2.11	BYTEORDER_SWAPW16	375
26.24.2.12	BYTEORDER_SWAPW32	375
26.24.2.13	C74_BIG_ENDIAN	375
26.24.2.14	C74_LITTLE_ENDIAN	376
26.25	Extending expr	377
26.25.1	Detailed Description	378
26.25.2	Enumeration Type Documentation	378
26.25.2.1	e_max_expr_types	378
26.25.3	Function Documentation	379
26.25.3.1	expr_eval	379
26.25.3.2	expr_new	379
26.26	Table Access	381
26.26.1	Detailed Description	381
26.26.2	Function Documentation	381
26.26.2.1	table_dirty	381
26.26.2.2	table_get	381
26.27	Text Editor Windows	383
26.28	Presets	384
26.28.1	Detailed Description	384
26.28.2	Function Documentation	385
26.28.2.1	preset_int	385
26.28.2.2	preset_set	385
26.28.2.3	preset_store	386
26.29	Event and File Serial Numbers	387
26.29.1	Detailed Description	387
26.29.2	Using Event Serial Numbers	387
26.29.3	Function Documentation	387
26.29.3.1	evnum_get	387
26.29.3.2	serialno	388
26.30	Loading Max Files	389
26.30.1	Detailed Description	389
26.30.2	Function Documentation	389
26.30.2.1	fileload	389
26.30.2.2	intload	390

26.30.2.3 readtohandle . . . . .	390
26.30.2.4 stringload . . . . .	390
26.31 Monitors and Displays . . . . .	392
26.31.1 Detailed Description . . . . .	392
26.31.2 Function Documentation . . . . .	392
26.31.2.1 jmonitor_getdisplayrect . . . . .	392
26.31.2.2 jmonitor_getdisplayrect_foralldisplays . . . . .	392
26.31.2.3 jmonitor_getdisplayrect_forpoint . . . . .	393
26.31.2.4 jmonitor_getnumdisplays . . . . .	393
26.32 Windows . . . . .	394
26.32.1 Function Documentation . . . . .	394
26.32.1.1 jwind_getactive . . . . .	394
26.32.1.2 jwind_getat . . . . .	394
26.32.1.3 jwind_getcount . . . . .	394
26.33 Mouse and Keyboard . . . . .	395
26.33.1 Enumeration Type Documentation . . . . .	396
26.33.1.1 t_jmouse_cursortype . . . . .	396
26.33.1.2 t_modifiers . . . . .	397
26.33.2 Function Documentation . . . . .	397
26.33.2.1 jkeyboard_getcurrentmodifiers . . . . .	397
26.33.2.2 jkeyboard_getcurrentmodifiers_realtime . . . . .	397
26.33.2.3 jmouse_getposition_global . . . . .	398
26.33.2.4 jmouse_setcursor . . . . .	398
26.33.2.5 jmouse_setposition_box . . . . .	398
26.33.2.6 jmouse_setposition_global . . . . .	398
26.33.2.7 jmouse_setposition_view . . . . .	398
26.34 Example Projects . . . . .	399
26.35 MSP . . . . .	402
26.35.1 Define Documentation . . . . .	404
26.35.1.1 PI . . . . .	404
26.35.1.2 PIOVERTWO . . . . .	404
26.35.1.3 TWOPI . . . . .	404
26.35.2 Typedef Documentation . . . . .	404
26.35.2.1 t_float . . . . .	404
26.35.2.2 t_int . . . . .	405
26.35.2.3 t_perfroutine . . . . .	405

26.35.2.4 t_sample . . . . .	405
26.35.3 Enumeration Type Documentation . . . . .	405
26.35.3.1 "@20 . . . . .	405
26.35.4 Function Documentation . . . . .	405
26.35.4.1 class_dspinit . . . . .	405
26.35.4.2 class_dspinitbox . . . . .	406
26.35.4.3 class_dspinitjbox . . . . .	406
26.35.4.4 dsp_add . . . . .	406
26.35.4.5 dsp_addv . . . . .	406
26.35.4.6 sys_getblksize . . . . .	407
26.35.4.7 sys_getdspstate . . . . .	407
26.35.4.8 sys_getmaxblksize . . . . .	407
26.35.4.9 sys_getsr . . . . .	407
26.35.4.10 dsp_free . . . . .	407
26.35.4.11 dsp_setup . . . . .	408
26.36 Buffers . . . . .	409
26.36.1 Detailed Description . . . . .	409
26.37 PFFT . . . . .	411
26.37.1 Detailed Description . . . . .	411
26.38 Poly . . . . .	412
26.39 Objects . . . . .	413
26.39.1 Detailed Description . . . . .	416
26.39.2 Define Documentation . . . . .	417
26.39.2.1 MAXARG . . . . .	417
26.39.3 Function Documentation . . . . .	417
26.39.3.1 classname_openhelp . . . . .	417
26.39.3.2 classname_openquery . . . . .	417
26.39.3.3 classname_openrefpage . . . . .	417
26.39.3.4 newobject_fromdictionary . . . . .	417
26.39.3.5 newobject_sprintf . . . . .	418
26.39.3.6 object_alloc . . . . .	419
26.39.3.7 object_attach . . . . .	419
26.39.3.8 object_attach_byptr . . . . .	420
26.39.3.9 object_attach_byptr_register . . . . .	420
26.39.3.10 object_class . . . . .	420
26.39.3.11 object_classname . . . . .	421

26.39.3.12object_classname_compare . . . . .	421
26.39.3.13object_detach . . . . .	421
26.39.3.14object_detach_byptr . . . . .	422
26.39.3.15object_dictionaryarg . . . . .	422
26.39.3.16object_findregistered . . . . .	422
26.39.3.17object_findregisteredbyptr . . . . .	423
26.39.3.18object_free . . . . .	423
26.39.3.19object_getmethod . . . . .	423
26.39.3.20object_getvalueof . . . . .	423
26.39.3.21object_method . . . . .	424
26.39.3.22object_method_char . . . . .	425
26.39.3.23object_method_char_array . . . . .	425
26.39.3.24object_method_double . . . . .	426
26.39.3.25object_method_double_array . . . . .	426
26.39.3.26object_method_float . . . . .	426
26.39.3.27object_method_float_array . . . . .	427
26.39.3.28object_method_format . . . . .	427
26.39.3.29object_method_long . . . . .	428
26.39.3.30object_method_long_array . . . . .	428
26.39.3.31object_method_obj . . . . .	428
26.39.3.32object_method_obj_array . . . . .	429
26.39.3.33object_method_parse . . . . .	429
26.39.3.34object_method_sym . . . . .	430
26.39.3.35object_method_sym_array . . . . .	430
26.39.3.36object_method_typed . . . . .	430
26.39.3.37object_method_typedfun . . . . .	431
26.39.3.38object_new . . . . .	431
26.39.3.39object_new_typed . . . . .	432
26.39.3.40object_notify . . . . .	432
26.39.3.41object_obex_dumpout . . . . .	433
26.39.3.42object_obex_lookup . . . . .	433
26.39.3.43object_obex_store . . . . .	433
26.39.3.44object_openhelp . . . . .	434
26.39.3.45object_openquery . . . . .	434
26.39.3.46object_openrefpage . . . . .	434
26.39.3.47object_register . . . . .	434

26.39.3.48	object_setvalueof	435
26.39.3.49	object_unregister	435
26.40	Patcher	437
26.40.1	Detailed Description	438
26.40.2	Typedef Documentation	438
26.40.2.1	t_box	438
26.40.2.2	t_patcher	438
26.40.3	Enumeration Type Documentation	438
26.40.3.1	"@3	438
26.41	jpatcher	439
26.41.1	Detailed Description	441
26.41.2	Function Documentation	441
26.41.2.1	jpatcher_deleteobj	441
26.41.2.2	jpatcher_get_bgcolor	442
26.41.2.3	jpatcher_get_bghidden	442
26.41.2.4	jpatcher_get_bglocked	442
26.41.2.5	jpatcher_get_box	442
26.41.2.6	jpatcher_get_count	443
26.41.2.7	jpatcher_get_currentfileversion	443
26.41.2.8	jpatcher_get_default_fontface	443
26.41.2.9	jpatcher_get_default_fontname	443
26.41.2.10	jpatcher_get_default_fontsize	443
26.41.2.11	jpatcher_get_defrect	444
26.41.2.12	jpatcher_get_dirty	444
26.41.2.13	jpatcher_get_editing_bgcolor	444
26.41.2.14	jpatcher_get_fghidden	444
26.41.2.15	jpatcher_get_filename	445
26.41.2.16	jpatcher_get_filepath	445
26.41.2.17	jpatcher_get_fileversion	445
26.41.2.18	jpatcher_get_firstline	445
26.41.2.19	jpatcher_get_firstobject	446
26.41.2.20	jpatcher_get_firstview	446
26.41.2.21	jpatcher_get_gridsize	446
26.41.2.22	jpatcher_get_lastobject	446
26.41.2.23	jpatcher_get_name	447
26.41.2.24	jpatcher_get_parentpatcher	447

26.41.2.25	patcher_get_presentation	447
26.41.2.26	patcher_get_rect	447
26.41.2.27	patcher_get_title	448
26.41.2.28	patcher_get_toppatcher	448
26.41.2.29	patcher_is_patcher	448
26.41.2.30	patcher_set_bgcolor	448
26.41.2.31	patcher_set_bghidden	449
26.41.2.32	patcher_set_bglocked	449
26.41.2.33	patcher_set_defrect	449
26.41.2.34	patcher_set_dirty	449
26.41.2.35	patcher_set_editing_bgcolor	450
26.41.2.36	patcher_set_fghidden	450
26.41.2.37	patcher_set_gridsize	450
26.41.2.38	patcher_set_locked	450
26.41.2.39	patcher_set_presentation	451
26.41.2.40	patcher_set_rect	451
26.41.2.41	patcher_set_title	451
26.41.2.42	patcher_uniqueboxname	451
26.42	jbox	452
26.42.1	Detailed Description	457
26.42.2	Define Documentation	458
26.42.2.1	JBOX_NOINSPECTFIRSTIN	458
26.42.3	Enumeration Type Documentation	458
26.42.3.1	"@ 17	458
26.42.3.2	HitTestResult	458
26.42.4	Function Documentation	458
26.42.4.1	jbox_free	458
26.42.4.2	jbox_get_annotation	459
26.42.4.3	jbox_get_background	459
26.42.4.4	jbox_get_canhilite	459
26.42.4.5	jbox_get_color	459
26.42.4.6	jbox_get_drawfirstin	460
26.42.4.7	jbox_get_drawinlast	460
26.42.4.8	jbox_get_fontname	460
26.42.4.9	jbox_get_fontsize	460
26.42.4.10	jbox_get_growboth	461

26.42.4.1	<a href="#">lbox_get_growy</a>	461
26.42.4.1	<a href="#">lbox_get_hidden</a>	461
26.42.4.1	<a href="#">lbox_get_hint</a>	461
26.42.4.1	<a href="#">lbox_get_hintstring</a>	462
26.42.4.1	<a href="#">lbox_get_id</a>	462
26.42.4.1	<a href="#">lbox_get_ignoreclick</a>	462
26.42.4.1	<a href="#">lbox_get_maxclass</a>	462
26.42.4.1	<a href="#">lbox_get_nextobject</a>	463
26.42.4.1	<a href="#">lbox_get_nogrow</a>	463
26.42.4.1	<a href="#">lbox_get_object</a>	463
26.42.4.1	<a href="#">lbox_get_outline</a>	463
26.42.4.1	<a href="#">lbox_get_patcher</a>	464
26.42.4.1	<a href="#">lbox_get_patching_position</a>	464
26.42.4.1	<a href="#">lbox_get_patching_rect</a>	464
26.42.4.1	<a href="#">lbox_get_patching_size</a>	464
26.42.4.1	<a href="#">lbox_get_presentation</a>	465
26.42.4.1	<a href="#">lbox_get_presentation_position</a>	465
26.42.4.1	<a href="#">lbox_get_presentation_rect</a>	465
26.42.4.1	<a href="#">lbox_get_presentation_size</a>	465
26.42.4.1	<a href="#">lbox_get_prevobject</a>	466
26.42.4.1	<a href="#">lbox_get_rect_for_sym</a>	466
26.42.4.1	<a href="#">lbox_get_rect_for_view</a>	466
26.42.4.1	<a href="#">lbox_get_textfield</a>	466
26.42.4.1	<a href="#">lbox_get_varname</a>	467
26.42.4.1	<a href="#">lbox_new</a>	467
26.42.4.1	<a href="#">lbox_notify</a>	467
26.42.4.1	<a href="#">lbox_ready</a>	467
26.42.4.1	<a href="#">lbox_redraw</a>	468
26.42.4.1	<a href="#">lbox_set_annotation</a>	468
26.42.4.1	<a href="#">lbox_set_background</a>	468
26.42.4.1	<a href="#">lbox_set_color</a>	468
26.42.4.1	<a href="#">lbox_set_fontname</a>	469
26.42.4.1	<a href="#">lbox_set_fontsize</a>	469
26.42.4.1	<a href="#">lbox_set_hidden</a>	469
26.42.4.1	<a href="#">lbox_set_hint</a>	469
26.42.4.1	<a href="#">lbox_set_hintstring</a>	470

26.42.4.47	<a href="#">jbox_set_ignoreclick</a>	470
26.42.4.48	<a href="#">jbox_set_outline</a>	470
26.42.4.49	<a href="#">jbox_set_patching_position</a>	470
26.42.4.50	<a href="#">jbox_set_patching_rect</a>	471
26.42.4.51	<a href="#">jbox_set_patching_size</a>	471
26.42.4.52	<a href="#">jbox_set_position</a>	471
26.42.4.53	<a href="#">jbox_set_presentation</a>	471
26.42.4.54	<a href="#">jbox_set_presentation_position</a>	472
26.42.4.55	<a href="#">jbox_set_presentation_rect</a>	472
26.42.4.56	<a href="#">jbox_set_presentation_size</a>	472
26.42.4.57	<a href="#">jbox_set_rect</a>	472
26.42.4.58	<a href="#">jbox_set_rect_for_sym</a>	473
26.42.4.59	<a href="#">jbox_set_rect_for_view</a>	473
26.42.4.60	<a href="#">jbox_set_size</a>	473
26.42.4.61	<a href="#">jbox_set_varname</a>	473
26.43	<a href="#">jpatchline</a>	474
26.43.1	Detailed Description	475
26.43.2	Function Documentation	475
26.43.2.1	<a href="#">jpatchline_get_box1</a>	475
26.43.2.2	<a href="#">jpatchline_get_box2</a>	475
26.43.2.3	<a href="#">jpatchline_get_color</a>	475
26.43.2.4	<a href="#">jpatchline_get_endpoint</a>	475
26.43.2.5	<a href="#">jpatchline_get_hidden</a>	476
26.43.2.6	<a href="#">jpatchline_get_inletnum</a>	476
26.43.2.7	<a href="#">jpatchline_get_nextline</a>	476
26.43.2.8	<a href="#">jpatchline_get_nummidpoints</a>	476
26.43.2.9	<a href="#">jpatchline_get_outletnum</a>	477
26.43.2.10	<a href="#">jpatchline_get_startpoint</a>	477
26.43.2.11	<a href="#">jpatchline_set_color</a>	477
26.43.2.12	<a href="#">jpatchline_set_hidden</a>	477
26.44	<a href="#">jpatcherview</a>	478
26.44.1	Detailed Description	479
26.44.2	Function Documentation	479
26.44.2.1	<a href="#">patcherview_findpatcherview</a>	479
26.44.2.2	<a href="#">patcherview_get_jgraphics</a>	479
26.44.2.3	<a href="#">patcherview_get_locked</a>	479



26.44.2.4 patchview_get_nextview . . . . .	480
26.44.2.5 patchview_get_patcher . . . . .	480
26.44.2.6 patchview_get_presentation . . . . .	480
26.44.2.7 patchview_get_rect . . . . .	480
26.44.2.8 patchview_get_visible . . . . .	481
26.44.2.9 patchview_get_zoomfactor . . . . .	481
26.44.2.10 patchview_set_locked . . . . .	481
26.44.2.11 patchview_set_presentation . . . . .	481
26.44.2.12 patchview_set_rect . . . . .	482
26.44.2.13 patchview_set_visible . . . . .	482
26.44.2.14 patchview_set_zoomfactor . . . . .	482
26.45 Timing . . . . .	483
26.46 Clocks . . . . .	484
26.46.1 Detailed Description . . . . .	485
26.46.2 Using Clocks . . . . .	485
26.46.3 Scheduling with setclock Objects . . . . .	486
26.46.3.1 Using the setclock Object Routines . . . . .	487
26.46.4 Creating Schedulers . . . . .	488
26.46.5 Function Documentation . . . . .	488
26.46.5.1 clock_delay . . . . .	488
26.46.5.2 clock_fdelay . . . . .	488
26.46.5.3 clock_gettime . . . . .	489
26.46.5.4 clock_new . . . . .	489
26.46.5.5 clock_unset . . . . .	489
26.46.5.6 gettimeofday . . . . .	489
26.46.5.7 scheduler_gettime . . . . .	490
26.46.5.8 scheduler_new . . . . .	490
26.46.5.9 scheduler_run . . . . .	490
26.46.5.10 scheduler_set . . . . .	490
26.46.5.11 scheduler_settime . . . . .	491
26.46.5.12 setclock_delay . . . . .	491
26.46.5.13 setclock_fdelay . . . . .	491
26.46.5.14 setclock_gettime . . . . .	491
26.46.5.15 setclock_gettime . . . . .	492
26.46.5.16 setclock_unset . . . . .	492
26.46.5.17 systimer_gettime . . . . .	492

26.47	Qelems	493
26.47.1	Detailed Description	493
26.47.2	Function Documentation	494
26.47.2.1	qelem_free	494
26.47.2.2	qelem_front	494
26.47.2.3	qelem_new	494
26.47.2.4	qelem_set	495
26.47.2.5	qelem_unset	495
26.48	Systime API	496
26.48.1	Detailed Description	497
26.48.2	Enumeration Type Documentation	497
26.48.2.1	e_max_dateflags	497
26.48.3	Function Documentation	497
26.48.3.1	sysdateformat_formatdatetime	497
26.48.3.2	sysdateformat_strftimetodatetime	497
26.48.3.3	systime_datetime	498
26.48.3.4	systime_datetoseconds	498
26.48.3.5	systime_ms	498
26.48.3.6	systime_seconds	498
26.48.3.7	systime_secondstodate	498
26.48.3.8	systime_ticks	499
26.49	ITM Time Objects	500
26.49.1	Detailed Description	503
26.49.2	Enumeration Type Documentation	503
26.49.2.1	"@2	503
26.49.3	Function Documentation	504
26.49.3.1	class_time_addattr	504
26.49.3.2	itm_barbeatunitstoticks	504
26.49.3.3	itm_dereference	504
26.49.3.4	itm_dump	505
26.49.3.5	itm_getglobal	505
26.49.3.6	itm_getname	505
26.49.3.7	itm_getnamed	505
26.49.3.8	itm_getresolution	505
26.49.3.9	itm_getstate	506
26.49.3.10	itm_getticks	506

26.49.3.1	litm_gettime	506
26.49.3.2	litm_gettimesignature	506
26.49.3.3	litm_isunitfixed	507
26.49.3.4	litm_mstosamps	507
26.49.3.5	litm_mstoticks	507
26.49.3.6	litm_pause	507
26.49.3.7	litm_reference	508
26.49.3.8	litm_resume	508
26.49.3.9	litm_sampstoms	508
26.49.3.10	litm_setresolution	508
26.49.3.2	litm_settimesignature	508
26.49.3.22	litm_tickstobarbeatunits	509
26.49.3.23	litm_tickstoms	509
26.49.3.24	time_calcquantize	509
26.49.3.25	time_getitm	509
26.49.3.26	time_getms	510
26.49.3.27	time_getnamed	510
26.49.3.28	time_getphase	510
26.49.3.29	time_getticks	510
26.49.3.30	time_isfixedunit	511
26.49.3.31	time_listen	511
26.49.3.32	time_new	511
26.49.3.33	time_now	511
26.49.3.34	time_schedule	512
26.49.3.35	time_schedule_limit	512
26.49.3.36	time_setclock	512
26.49.3.37	time_setvalue	512
26.49.3.38	time_stop	512
26.49.3.39	time_tick	513
26.49.4	Variable Documentation	513
26.49.4.1	t_itm	513
26.49.4.2	t_timeobject	513
26.50	Threads	514
26.50.1	Detailed Description	516
26.50.2	Define Documentation	516
26.50.2.1	ATOMIC_DECREMENT	516

26.50.2.2	ATOMIC_INCREMENT	516
26.50.3	Enumeration Type Documentation	516
26.50.3.1	e_max_systhread_mutex_flags	516
26.50.4	Function Documentation	517
26.50.4.1	defer	517
26.50.4.2	defer_low	517
26.50.4.3	isr	518
26.50.4.4	schedule	518
26.50.4.5	schedule_delay	519
26.50.4.6	systhread_create	519
26.50.4.7	systhread_exit	520
26.50.4.8	systhread_ismainthread	520
26.50.4.9	systhread_istimerthread	520
26.50.4.10	systhread_join	520
26.50.4.11	systhread_self	521
26.50.4.12	systhread_sleep	521
26.50.4.13	systhread_terminate	521
26.51	Critical Regions	522
26.51.1	Detailed Description	522
26.51.2	Function Documentation	523
26.51.2.1	critical_enter	523
26.51.2.2	critical_exit	524
26.51.2.3	critical_free	524
26.51.2.4	critical_new	524
26.51.2.5	critical_tryenter	524
26.52	Mutexes	525
26.52.1	Detailed Description	525
26.52.2	Function Documentation	525
26.52.2.1	systhread_mutex_free	525
26.52.2.2	systhread_mutex_lock	526
26.52.2.3	systhread_mutex_new	526
26.52.2.4	systhread_mutex_newlock	526
26.52.2.5	systhread_mutex_trylock	527
26.52.2.6	systhread_mutex_unlock	527
26.53	User Interface	528
26.54	Graphics	529

26.54.1 Detailed Description	534
26.54.2 Define Documentation	534
26.54.2.1 JGRAPHICS_2PI	534
26.54.2.2 JGRAPHICS_3PIOVER2	534
26.54.2.3 JGRAPHICS_PI	534
26.54.2.4 JGRAPHICS_PIOVER2	534
26.54.3 Enumeration Type Documentation	534
26.54.3.1 t_jgraphics_fileformat	534
26.54.3.2 t_jgraphics_format	535
26.54.3.3 t_jgraphics_text_justification	535
26.54.4 Function Documentation	535
26.54.4.1 jgraphics_append_path	535
26.54.4.2 jgraphics_arc	535
26.54.4.3 jgraphics_arc_negative	536
26.54.4.4 jgraphics_close_path	536
26.54.4.5 jgraphics_copy_path	536
26.54.4.6 jgraphics_curve_to	536
26.54.4.7 jgraphics_destroy	537
26.54.4.8 jgraphics_device_to_user	537
26.54.4.9 jgraphics_ellipse	537
26.54.4.10 jgraphics_font_extents	537
26.54.4.11 jgraphics_get_current_point	537
26.54.4.12 jgraphics_getfiletypes	538
26.54.4.13 jgraphics_line_to	538
26.54.4.14 jgraphics_move_to	538
26.54.4.15 jgraphics_new_path	539
26.54.4.16 jgraphics_oval	539
26.54.4.17 jgraphics_ovalarc	539
26.54.4.18 jgraphics_path_destroy	539
26.54.4.19 jgraphics_path_roundcorners	540
26.54.4.20 jgraphics_position_one_rect_near_another_rect_but_keep_inside_a_- third_rect	540
26.54.4.21 jgraphics_rectangle	540
26.54.4.22 jgraphics_rectangle_rounded	540
26.54.4.23 jgraphics_rectcontainsrect	541
26.54.4.24 jgraphics_rectintersectsrect	541
26.54.4.25 jgraphics_reference	541

26.54.4.26	jgraphics_rel_curve_to	541
26.54.4.27	jgraphics_rel_line_to	542
26.54.4.28	jgraphics_rel_move_to	542
26.54.4.29	jgraphics_round	542
26.54.4.30	jgraphics_select_font_face	542
26.54.4.31	jgraphics_select_jfont	542
26.54.4.32	jgraphics_set_font_size	543
26.54.4.33	jgraphics_set_underline	543
26.54.4.34	jgraphics_show_text	543
26.54.4.35	jgraphics_system_canantialiastexttotransparentbg	543
26.54.4.36	jgraphics_text_measure	543
26.54.4.37	jgraphics_text_measure_wrapped	544
26.54.4.38	jgraphics_user_to_device	544
26.55	JSurface	545
26.55.1	Detailed Description	546
26.55.2	Function Documentation	547
26.55.2.1	jgraphics_create	547
26.55.2.2	jgraphics_get_resource_data	547
26.55.2.3	jgraphics_image_surface_clear	547
26.55.2.4	jgraphics_image_surface_create	548
26.55.2.5	jgraphics_image_surface_create_for_data	548
26.55.2.6	jgraphics_image_surface_create_from_file	548
26.55.2.7	jgraphics_image_surface_create_from_filedata	549
26.55.2.8	jgraphics_image_surface_create_from_resource	549
26.55.2.9	jgraphics_image_surface_create_referenced	549
26.55.2.10	jgraphics_image_surface_draw	550
26.55.2.11	jgraphics_image_surface_draw_fast	550
26.55.2.12	jgraphics_image_surface_get_height	550
26.55.2.13	jgraphics_image_surface_get_pixel	551
26.55.2.14	jgraphics_image_surface_get_width	551
26.55.2.15	jgraphics_image_surface_lockpixels	551
26.55.2.16	jgraphics_image_surface_lockpixels_readonly	551
26.55.2.17	jgraphics_image_surface_scroll	552
26.55.2.18	jgraphics_image_surface_set_pixel	552
26.55.2.19	jgraphics_image_surface_unlockpixels	552
26.55.2.20	jgraphics_image_surface_unlockpixels_readonly	553

26.55.2.2	<a href="#">jgraphics_image_surface_writepng</a>	553
26.55.2.2	<a href="#">jgraphics_surface_destroy</a>	553
26.55.2.2	<a href="#">jgraphics_surface_reference</a>	553
26.55.2.2	<a href="#">jgraphics_write_image_surface_to_filedata</a>	554
26.56	Scalable Vector Graphics	555
26.56.1	Function Documentation	555
26.56.1.1	<a href="#">jsvg_create_from_file</a>	555
26.56.1.2	<a href="#">jsvg_create_from_resource</a>	555
26.56.1.3	<a href="#">jsvg_create_from_xmlstring</a>	556
26.56.1.4	<a href="#">jsvg_destroy</a>	556
26.56.1.5	<a href="#">jsvg_get_size</a>	556
26.56.1.6	<a href="#">jsvg_render</a>	556
26.57	JFont	557
26.57.1	Enumeration Type Documentation	558
26.57.1.1	<a href="#">t_jgraphics_font_slant</a>	558
26.57.1.2	<a href="#">t_jgraphics_font_weight</a>	558
26.57.2	Function Documentation	558
26.57.2.1	<a href="#">jbox_get_font_slant</a>	558
26.57.2.2	<a href="#">jbox_get_font_weight</a>	559
26.57.2.3	<a href="#">jfont_create</a>	559
26.57.2.4	<a href="#">jfont_destroy</a>	559
26.57.2.5	<a href="#">jfont_extents</a>	559
26.57.2.6	<a href="#">jfont_getfontlist</a>	560
26.57.2.7	<a href="#">jfont_reference</a>	560
26.57.2.8	<a href="#">jfont_set_font_size</a>	560
26.57.2.9	<a href="#">jfont_set_underline</a>	560
26.57.2.10	<a href="#">jfont_text_measure</a>	560
26.57.2.11	<a href="#">jfont_text_measure_wrapped</a>	561
26.58	JGraphics Matrix Transformations	562
26.58.1	Detailed Description	563
26.58.2	Function Documentation	563
26.58.2.1	<a href="#">jgraphics_matrix_init</a>	563
26.58.2.2	<a href="#">jgraphics_matrix_init_identity</a>	563
26.58.2.3	<a href="#">jgraphics_matrix_init_rotate</a>	563
26.58.2.4	<a href="#">jgraphics_matrix_init_scale</a>	564
26.58.2.5	<a href="#">jgraphics_matrix_init_translate</a>	564

26.58.2.6	jgraphics_matrix_invert	564
26.58.2.7	jgraphics_matrix_multiply	564
26.58.2.8	jgraphics_matrix_rotate	564
26.58.2.9	jgraphics_matrix_scale	565
26.58.2.10	jgraphics_matrix_transform_point	565
26.58.2.11	jgraphics_matrix_translate	565
26.59	JPattern	566
26.59.1	Detailed Description	566
26.60	Colors	567
26.60.1	Function Documentation	567
26.60.1.1	atoms_to_jrgba	567
26.60.1.2	jrgba_attr_get	568
26.60.1.3	jrgba_attr_set	568
26.60.1.4	jrgba_compare	568
26.60.1.5	jrgba_copy	569
26.60.1.6	jrgba_set	569
26.60.1.7	jrgba_to_atoms	569
26.61	TextField	570
26.61.1	Detailed Description	572
26.61.2	Function Documentation	572
26.61.2.1	textfield_get_autoscroll	572
26.61.2.2	textfield_get_bgcolor	572
26.61.2.3	textfield_get_editonlick	572
26.61.2.4	textfield_get_emptytext	572
26.61.2.5	textfield_get_noactivate	573
26.61.2.6	textfield_get_owner	573
26.61.2.7	textfield_get_readonly	573
26.61.2.8	textfield_get_selectallonedit	573
26.61.2.9	textfield_get_textcolor	574
26.61.2.10	textfield_get_textmargins	574
26.61.2.11	textfield_get_underline	574
26.61.2.12	textfield_get_useellipsis	574
26.61.2.13	textfield_get_wantsreturn	575
26.61.2.14	textfield_get_wantstab	575
26.61.2.15	textfield_get_wordwrap	575
26.61.2.16	textfield_set_autoscroll	575



26.61.2.17	<a href="#">textfield_set_bgcolor</a>	576
26.61.2.18	<a href="#">textfield_set_editonclick</a>	576
26.61.2.19	<a href="#">textfield_set_emptytext</a>	576
26.61.2.20	<a href="#">textfield_set_noactivate</a>	576
26.61.2.21	<a href="#">textfield_set_readonly</a>	577
26.61.2.22	<a href="#">textfield_set_selectallonedit</a>	577
26.61.2.23	<a href="#">textfield_set_textcolor</a>	577
26.61.2.24	<a href="#">textfield_set_textmargins</a>	577
26.61.2.25	<a href="#">textfield_set_underline</a>	578
26.61.2.26	<a href="#">textfield_set_useellipsis</a>	578
26.61.2.27	<a href="#">textfield_set_wantsreturn</a>	578
26.61.2.28	<a href="#">textfield_set_wantstab</a>	578
26.61.2.29	<a href="#">textfield_set_wordwrap</a>	579
26.62	<a href="#">TextLayout</a>	580
26.62.1	<a href="#">Detailed Description</a>	581
26.62.2	<a href="#">Enumeration Type Documentation</a>	581
26.62.2.1	<a href="#">t_jgraphics_textlayout_flags</a>	581
26.62.3	<a href="#">Function Documentation</a>	581
26.62.3.1	<a href="#">jtextlayout_create</a>	581
26.62.3.2	<a href="#">jtextlayout_destroy</a>	581
26.62.3.3	<a href="#">jtextlayout_draw</a>	581
26.62.3.4	<a href="#">jtextlayout_getchar</a>	582
26.62.3.5	<a href="#">jtextlayout_getcharbox</a>	582
26.62.3.6	<a href="#">jtextlayout_getnumchars</a>	582
26.62.3.7	<a href="#">jtextlayout_measure</a>	582
26.62.3.8	<a href="#">jtextlayout_set</a>	583
26.62.3.9	<a href="#">jtextlayout_settextcolor</a>	583
26.62.3.10	<a href="#">jtextlayout_withbgcolor</a>	583
26.63	<a href="#">Popup Menus</a>	584
26.63.1	<a href="#">Detailed Description</a>	585
26.63.2	<a href="#">Function Documentation</a>	585
26.63.2.1	<a href="#">jpopupmenu_additem</a>	585
26.63.2.2	<a href="#">jpopupmenu_addseparator</a>	585
26.63.2.3	<a href="#">jpopupmenu_addsubmenu</a>	585
26.63.2.4	<a href="#">jpopupmenu_clear</a>	585
26.63.2.5	<a href="#">jpopupmenu_create</a>	586

26.63.2.6 jpopupmenu_destroy . . . . .	586
26.63.2.7 jpopupmenu_popup . . . . .	586
26.63.2.8 jpopupmenu_popup_abovebox . . . . .	586
26.63.2.9 jpopupmenu_popup_nearbox . . . . .	587
26.63.2.10 jpopupmenu_setcolors . . . . .	587
26.63.2.11 jpopupmenu_setfont . . . . .	587
26.64 Box Layer . . . . .	588
26.64.1 Detailed Description . . . . .	588
26.64.2 Function Documentation . . . . .	589
26.64.2.1 jbox_end_layer . . . . .	589
26.64.2.2 jbox_invalidate_layer . . . . .	589
26.64.2.3 jbox_paint_layer . . . . .	589
26.64.2.4 jbox_start_layer . . . . .	590
26.65 DataView . . . . .	591
26.65.1 Detailed Description . . . . .	591
26.65.2 Function Documentation . . . . .	592
26.65.2.1 jdataview_getclient . . . . .	592
26.65.2.2 jdataview_new . . . . .	592
26.65.2.3 jdataview_setclient . . . . .	592
26.66 Unicode . . . . .	593
26.66.1 Detailed Description . . . . .	593
26.66.2 Character Encodings . . . . .	593
26.66.2.1 Example Usage . . . . .	594
26.66.3 Function Documentation . . . . .	594
26.66.3.1 charset_convert . . . . .	594
26.66.3.2 charset_unicodetoutf8 . . . . .	594
26.66.3.3 charset_utf8_count . . . . .	595
26.66.3.4 charset_utf8_offset . . . . .	595
26.66.3.5 charset_utf8tounicode . . . . .	595
<b>27 Data Structure Documentation</b>	<b>597</b>
27.1 Ex_ex Struct Reference . . . . .	597
27.1.1 Detailed Description . . . . .	597
27.2 t_atom Struct Reference . . . . .	598
27.2.1 Detailed Description . . . . .	598
27.3 t_atomarray Struct Reference . . . . .	599
27.3.1 Detailed Description . . . . .	599

27.4	<a href="#">t_atombuf Struct Reference</a>	600
27.4.1	<a href="#">Detailed Description</a>	600
27.5	<a href="#">t_attr Struct Reference</a>	601
27.5.1	<a href="#">Detailed Description</a>	601
27.6	<a href="#">t_buffer Struct Reference</a>	602
27.6.1	<a href="#">Detailed Description</a>	604
27.7	<a href="#">t_celldesc Struct Reference</a>	605
27.7.1	<a href="#">Detailed Description</a>	605
27.8	<a href="#">t_charset_converter Struct Reference</a>	606
27.8.1	<a href="#">Detailed Description</a>	606
27.9	<a href="#">t_class Struct Reference</a>	607
27.9.1	<a href="#">Detailed Description</a>	607
27.10	<a href="#">t_datetime Struct Reference</a>	608
27.10.1	<a href="#">Detailed Description</a>	608
27.11	<a href="#">t_dictionary Struct Reference</a>	609
27.11.1	<a href="#">Detailed Description</a>	609
27.12	<a href="#">t_dictionary_entry Struct Reference</a>	611
27.12.1	<a href="#">Detailed Description</a>	611
27.13	<a href="#">t_expr Struct Reference</a>	612
27.13.1	<a href="#">Detailed Description</a>	612
27.14	<a href="#">t_fileinfo Struct Reference</a>	613
27.14.1	<a href="#">Detailed Description</a>	613
27.15	<a href="#">t_funbuff Struct Reference</a>	614
27.15.1	<a href="#">Detailed Description</a>	615
27.16	<a href="#">t_hashtab Struct Reference</a>	616
27.16.1	<a href="#">Detailed Description</a>	616
27.17	<a href="#">t_hashtab_entry Struct Reference</a>	617
27.17.1	<a href="#">Detailed Description</a>	617
27.18	<a href="#">t_indexmap Struct Reference</a>	618
27.18.1	<a href="#">Detailed Description</a>	618
27.19	<a href="#">t_indexmap_entry Struct Reference</a>	620
27.19.1	<a href="#">Detailed Description</a>	620
27.20	<a href="#">t_jbox Struct Reference</a>	621
27.20.1	<a href="#">Detailed Description</a>	621
27.21	<a href="#">t_jboxdrawparams Struct Reference</a>	622
27.21.1	<a href="#">Detailed Description</a>	622

27.22t_jcolumn Struct Reference . . . . .	623
27.22.1 Detailed Description . . . . .	626
27.23t_jdataview Struct Reference . . . . .	627
27.23.1 Detailed Description . . . . .	630
27.24t_jgraphics_font_extents Struct Reference . . . . .	631
27.24.1 Detailed Description . . . . .	631
27.25t_jmatrix Struct Reference . . . . .	632
27.25.1 Detailed Description . . . . .	632
27.26t_jrgb Struct Reference . . . . .	633
27.26.1 Detailed Description . . . . .	633
27.27t_jrgba Struct Reference . . . . .	634
27.27.1 Detailed Description . . . . .	634
27.28t_linklist Struct Reference . . . . .	635
27.28.1 Detailed Description . . . . .	635
27.29t_llelem Struct Reference . . . . .	636
27.29.1 Detailed Description . . . . .	636
27.30t_messlist Struct Reference . . . . .	637
27.30.1 Detailed Description . . . . .	637
27.31t_object Struct Reference . . . . .	638
27.31.1 Detailed Description . . . . .	638
27.32t_path Struct Reference . . . . .	639
27.32.1 Detailed Description . . . . .	639
27.33t_pathlink Struct Reference . . . . .	640
27.33.1 Detailed Description . . . . .	640
27.34t_pfftpub Struct Reference . . . . .	641
27.34.1 Detailed Description . . . . .	641
27.35t_privatesortrec Struct Reference . . . . .	643
27.35.1 Detailed Description . . . . .	644
27.36t_pt Struct Reference . . . . .	645
27.36.1 Detailed Description . . . . .	645
27.37t_pxjbox Struct Reference . . . . .	646
27.37.1 Detailed Description . . . . .	646
27.38t_pxobject Struct Reference . . . . .	647
27.38.1 Detailed Description . . . . .	647
27.39t_quickmap Struct Reference . . . . .	648
27.39.1 Detailed Description . . . . .	648

27.40t_rect Struct Reference . . . . .	649
27.40.1 Detailed Description . . . . .	649
27.41t_signal Struct Reference . . . . .	650
27.41.1 Detailed Description . . . . .	650
27.42t_size Struct Reference . . . . .	651
27.42.1 Detailed Description . . . . .	651
27.43t_string Struct Reference . . . . .	652
27.43.1 Detailed Description . . . . .	652
27.44t_symbol Struct Reference . . . . .	653
27.44.1 Detailed Description . . . . .	653
27.45t_symobject Struct Reference . . . . .	654
27.45.1 Detailed Description . . . . .	654
27.46t_tinyobject Struct Reference . . . . .	655
27.46.1 Detailed Description . . . . .	655
27.47t_zll Struct Reference . . . . .	656
27.47.1 Detailed Description . . . . .	656
27.48word Union Reference . . . . .	657
27.48.1 Detailed Description . . . . .	657



# 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.

### 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.



## **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 "sdk-build" two folder-levels up from the project. If you leave the arrangement of folders intact, sdk-build will be found in the MaxSDK folder.

We recommend that you add the sdk-build folder to your Max search path using the File Preferences window. This permits you to put the MaxSDK folder wherever you like and load the objects in Max after building them without copying them to your Cycling '74 folder.

## 2.2 Mac

Max external objects for the Mac are Mach-O bundles (folders that appear to be files) whose file-names must end with the .mxo extension. The example projects are in Xcode 3.x format. To download Xcode, you need to open a free Apple Developer account. For more information, visit <http://developer.apple.com/>

### 2.2.1 XCode Project Setup

The example projects are set up to have Development and Deployment build configurations. The Development configuration does not optimize and builds only for the target platform you are using (i.e., PPC on a PPC machine, Intel on an Intel machine). The Deployment configuration creates a universal binary and performs optimization.

The files required for this projects are included in the project folders with the except 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.

### 2.2.2 Linking and Frameworks

External objects use dynamic linking to access the API functions provided by the Max application. When an objects is loaded, calls to functions inside the application are resolved by the operating system to the correct memory address. Each external object Xcode project must reference MaxAPI.framework in order to link with the application. Frameworks are libraries that define the functions in the Max API. Due to the fact that "Max" could exist as an application, a standalone you create, or a library inside another application, the MaxAPI.framework does not actually contain the code to implement the functions of the Max API for external objects. It serves instead to isolate external objects from the specific library or application implementation that contains the real code.

Audio objects will link against MaxAudioAPI.framework and Jitter objects link against Jitter-API.framework. Unlike MaxAPI.framework, these frameworks are real libraries. 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.

## 2.3 Windows

Max external objects for Windows are Dynamic Link Libraries (DLLs) whose filenames must end with the .mxe extension. These DLLs will export a single function called "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\.

The example projects are in Visual C++ 2008 format. A free version of Visual C++ can be obtained from Microsoft at <http://www.microsoft.com/express/>. 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. By linking with this version of the C runtime library you won't have to worry about deployment issues due to dependencies your external may have on Microsoft's C Runtime. 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. If you prefer to use the Microsoft C Runtime you can do that by defining the C preprocessor macro MAXAPI\_USE\_MSRTC before including `ext.h`.

### 2.3.1 Compiling with Cygwin

It is also possible to compile Max external objects on Windows using Cygwin. The following steps show how to build the simplemax project from the MaxMSP SDK using Cygwin's gcc (Gnu Compiler Collection). This provides access to a high quality, free C compiler using the Cygwin Unix tools for Windows.

#### 2.3.1.1 Requirements

Install the following Cygwin packages. Feel free to add on any other Cygwin packages that strike your fancy. The Cygwin installer and more information can be found at <http://www.cygwin.com/>

- Base (ALL)
- Devel
  - binutils

- gcc "GCC Compiler"
- gcc-mingw "Mingw32 support headers and libraries for GCC"
- gcc-mingw-core "Mingw32 support headers and libraries for GCC"
- mingw-runtime

### 2.3.1.2 Build Steps

STEP 0: cd to the directory containing the minimum SDK example project

STEP 1:

```
gcc -c -mno-cygwin -DWIN_VERSION -DWIN_EXT_VERSION -I../c74support/max
-includes simplemax.c
```

Description of gcc arguments:

"-c" means compile.

"-mnocygwin" means use the Microsoft standard C libraries, instead of Cygwin standard C libraries. This step is important if you wish to distribute your extern to people that might not have Cygwin installed.

"-DWIN\_VERSION" and "-DWIN\_EXT\_VERSION" define these preprocessor definitions on the command line to guarantee that the header files and source code know it is being compiled for a Windows machine, instead of Macintosh.

"-I../c74support/max-includes" specifies an additional directory where the necessary headers files will be found.

"simplemax.c" is the compiler input.

STEP 2:

```
gcc -shared -mno-cygwin -o simplemax.mxe simplemax.o simplemax.def -L../
./c74support/max-includes -lMaxAPI
```

Description of gcc arguments:

"-shared" means link files to make a DLL.

"-mnocygwin" means use the Microsoft standard C libraries, instead of Cygwin standard C libraries. This step is important if you wish to distribute your extern to people that might not have Cygwin installed.

"-o simplemax.mxe" specifies the name of the output file.

"simplemax.o" and "simplemax.def" are the linker input. The .def file is necessary to ensure that the function main will be exported.

"-L../c74support/max-includes" specifies an additional directory where library files will be found.

"-lMaxAPI" means link to the MaxAPI.lib linker library for MaxAPI.dll.

STEP 3: copy your file to a directory in your search path. For example:

```
cp minimum.mxe c:\Program Files\Common Files\Cycling '74\myexterns\
```

### 2.3.1.3 Additional Notes

You can ignore the warning that main() does not return int. This message is harmless, and only relevant to applications, not shared libraries.

## 2.4 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.

### 2.4.1 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.

### 2.4.2 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.

## 2.5 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
#ifdef 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.



## **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 `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 `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 main()

int main()
{
    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;

    return 0;
}
```

[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 inlet. 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, 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);
    for (i = 0, ap = argv; i < argc; i++, ap++) {           // increment ap each
time to get to the next atom
        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.

## **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_object;
```

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_getftime(&time);
    post("instance %lx is executing at time %.2f", x, time);
}
```

3. 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);
```

4. 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);
```

5. 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);
```

5. 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 *arg
v)
{
    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 highpriority, `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:

#### 1) additional header files

After including `ext.h` and `ext_obex.h`, include `z_dsp.h`

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

#### 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;
```

#### 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, s
sizeof(t_mydspobject), 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_dsp, "dsp", A_CANT, 0);
```

#### 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");
```

#### 5) The dsp method and perform routine

The dsp method specifies the signal processing function your object defines along with its arguments. Your object's dsp 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 dsp method is declared as follows:

```
void mydspobject_dsp(t_mydspobject *x, t_signal **sp, short *count);
```

To add an entry to the DSP chain, your dsp method uses [dsp\\_add\(\)](#). The dsp method is passed an array of signals ([t\\_signal](#) pointers), which contain pointers to the actual sample memory your object's perform routine will be using for input and output. The array of signals starts with the inputs (from left to right), followed by the outputs. For example, if your object has two inputs (because your new instance routine called [dsp\\_setup\(x, 2\)](#)) and three outputs (because your new instance created three signal outlets), the signal array sp would contain five items as follows:

```
sp[0] // left input
sp[1] // right input
sp[2] // left output
sp[3] // middle output
sp[4] // right output
```

The [t\\_signal](#) data structure (defined in [z\\_dsp.h](#)), contains two important elements: the s\_n field, which is the size of the signal vector, and s\_vec, which is a pointer to an array of 32-bit floats containing the signal data. All t\_signals your object will receive have the same size. This size is not necessarily the same as the global MSP signal vector size, because your object might be inside a patcher within a poly~ object that defines its own size. Therefore it is important to use the s\_n field of a signal passed to your object's dsp method.

You can use a variety of strategies to pass arguments to your perform routine via [dsp\\_add\(\)](#). For simple unit generators that don't store any internal state between computing vectors, it is sufficient to pass the inputs, outputs, and vector size. For objects that need to store internal state between computing vectors such as filters or ramp generators, you will pass a pointer to your object, whose data structure should contain space to store this state. The plus1~ object does not need to store internal state. It passes the input, output, and vector size to its perform routine. The plus1~ dsp method is shown below:

```
void plus1_dsp(t_plus1 *x, t_signal **sp, short *count)
{
    dsp_add(plus1_perform, 3, sp[0]->s_vec, sp[1]->s_vec, sp[0]->s_n);
}
```

The first argument to [dsp\\_add\(\)](#) is your perform routine, followed by the number of additional arguments you wish to copy to the DSP chain, and then the arguments.

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. The design of the perform routine is somewhat unlike other Max methods. It receives a pointer to a piece of the DSP chain and it is expected to return the location of the next perform routine on the chain. The next location is determined by the number of arguments you specified for your perform routine with your call to [dsp\\_add\(\)](#). For example, if you will pass three arguments, you need to return w + 4.

Here is the plus1 perform routine:

```
t_int *plus1_perform(t_int *w)
{
    t_float *in, *out;
    int n;
```

```
in = (t_float *)w[1];          // get input signal vector
out = (t_float *)w[2];         // get output signal vector
n = (int)w[3];                 // vector size

while (n--)                    // perform calculation on all samples
    *out++ = *in++ + 1.;

return w + 4;                  // must return next DSP chain location
}
```

## 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;
    t_float f_sample;
} t_myfilter;
```

Then, in our dsp method (which has one input and one output), 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_dsp(t_myfilter *x, t_signal **sp, short *count)
{
    dsp_add(myfilter_perform, 4, x, sp[0]->s_vec, sp[1]->s_vec, sp[0]->s_n);

    x->f_sample = 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.

```
t_int *myfilter_perform(t_int *w)
{
    t_myfilter *x = (t_myfilter *)w[1];
    t_float *in = (t_float *)w[2];
    t_float *out = (t_float *)w[3];
    int n = (int)w[4];
    t_float samp = x->f_sample; // read from internal state
    t_float val;

    while (n--) {
        val = *in++;
        *out++ = (val + samp) * 0.5;
        samp = val;
    }
    x->f_sample = samp; // save to internal state

    return w + 5;
}
```

## 9.2 Observing Patcher Muting

The enable message to the thispatcher object, as well as the MSP mute object~ can be used to disable a subpatcher. If your object is at all computationally expensive in its perform routine, it should check to see whether it has been disabled. To do this, you'll need to pass a pointer to your object as one of the DSP chain arguments when calling [dsp\\_add\(\)](#). Here is a simple modification of our filter object's perform routine that checks to see if the object has been disabled.

```

t_int *myfilter_perform(t_int *w)
{
    t_myfilter *x = (t_myfilter *)w[1];
    t_float *in = (t_float *)w[2];
    t_float *out = (t_float *)w[3];
    int n = (int)w[4];
    t_float samp = x->f_sample; // read from internal state
    t_float val;

    if (x->f_obj.z_disabled) // check for object being disabled
        return w + 5;

    while (n--) {
        val = *in++;
        *out++ = (val + samp) * 0.5;
        samp = val;
    }
    x->f_sample = samp; // save to internal state

    return w + 5;
}

```

## 9.3 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_add()` altogether. The MSP signal operator objects (such as `+~` and `*~`) 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_add()`. The example below assumes that the second element in the signal and count arrays (`sp[1]`) is the right input:

```

if (count[1]) // signal connected
    dsp_add(mydspobject_twosigperform, 5, x, sp[0]->s_vec, sp[1]->s_vec,
sp[2]->s_vec, sp[0]->s_n);
else
    dsp_add(mydspobject_scalarperform, 4, x, sp[0]->s_vec, sp[2]->s_vec,
sp[0]->s_n);

```

The second option is to pass the value of the count array for a particular signal to the `perform` method, which can make the decision whether to use the signal value or a scalar value that has been stored inside 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 passed as the second argument on the DSP chain, and the right input signal vector is passed even if it not connected:

```

dsp_add(mydspobject_perform, 6, x, count[1], sp[0]->s_vec, sp[1]->s_vec,
sp[2]->s_vec, sp[0]->s_n);

```

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

```

t_int mydspobject_perform(t_int *w)

```

```
{
    t_mydspobject *x = (t_mydspobject *)w[1];
    int connected = (int)w[2];
    t_float *in = (t_float *)w[3];
    t_float *in2 = (t_float *)w[4];
    t_float *out = (t_float *)w[5];
    int n = (int)w[6];

    double in2value;

    // get scalar sample or use signal depending on whether signal is connect
ed

    in2value = connected? *in2 : x->m_scalarvalue;

    // do calculation here

    return w + 7;
}
```

## **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_ob;
    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_ob;
    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_ob);
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 `system_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 *)system_resizeptr((char *)x->m_data, size * sizeof
(long));
    else // first time alloc
        x->m_data = (long *)system_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 = system_ptrsize((char *)x->m_data) / sizeof(long); // calcul
ate 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 *se
nder, 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 header file descriptions of the set of [JGraphics](#) 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_out;                // 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 butt
                                // on 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 ove
                                // r 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 (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_uis
imp), 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 you 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 mousedrag method that performs this test and redraws the object if the "mouse inside" condition has changed from its previous state. The mousedrag 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 (using `jgraphics_ptinrect()`) to see if it is inside or outside the box.

```
void uisimp_mousedrag(t_uisimp *x, t_object *patcherview, t_pt pt, long modifiers)
{
    t_rect rect;

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

    // test to see if mouse is still inside the object, redraw if changed

    if (jgraphics_ptinrect(pt, rect)) {
        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 the `mouseleave` 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 `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)
{
    long filetype = 'TEXT', outtype;
    short numtypes = 1;
    char filename[512];
    short path;

    if (s == gensym("")) { // if no argument supplied, ask for file
        if (open_dialog(filename, &path, &outtype, &filetype, 1)) // no
n-zero: user cancelled
            return;
    } else {
        strcpy(filename, s->s_name); // must copy symbol before calling lo
catefile_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 = system_newhandle(0);
    sysfile_readtextfile(fh, texthandle, 0, 0);    // see flags explanation
    below
    post("the file has %ld characters", system_gethandlesize(texthandle));
    sysfile_close(fh);
    system_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 = system_newptr(size);

    // read in the file
    sysfile_read(fh, &size, buffer);

    sysfile_close(fh);

    // do something with data in buffer here

    system_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;

    obex_object_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_getparentpatcher()`. 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_getparentpatcher(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; 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;

patcher = 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 *x, 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;

patcher = 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(patcher, 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
fgcount	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



## 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

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", A_CANT, 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 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;
    }
}
```

## 14.3 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;
}
```

## 14.4 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 `edsave` message. If you return zero from your `edsave` 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 `edsave` message to trigger a recompile of the Javascript code. But it also returns 0 from its `edsave` method so that the text editor will update the script file. Except for the return value, the prototype of the `edsave` method is identical to the `edclose` method.

```
class_addmethod(c, (method)myobject_edsave, "edsave", A_CANT, 0);

long myobject_edsave(t_myobject *x, char **ht, long size)
{
    // do something with the text
    return 0;          // tell editor it can save the text
}
```

## 14.5 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"));
}
```

buffer~ support



## **Chapter 15**

# **Data Structures**

## 15.1 Linked Lists

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);
}
```

There are many more functions for operating on linked lists you can read about by exploring the [Linked List](#) function reference.

## 15.2 Hash Tables

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 as a [t\\_hashtab](#).



## **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 ("my-object"). For example, consider the patch shown below. On the left is a bang button connected to the left inlet. On the right is 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
    system_memcpy(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_ob;
    t_object *m_timeobj;
    void *m_outlet;
} _delay2simple;
```

Next, in your 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_tickstoms`. 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_tickstoms(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_TICKSONLY |
    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_TICKSONLY | 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**

# **JGraphics**

Coming soon! For now, refer to the [jgraphics.h](#) header file.

## **Chapter 20**

### **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.

## 20.1 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);	
quickref		obsolete, this is provided automatically now

## 20.2 Messages for Non-UI Objects

dblclick	void scripto_dblclick(t_scripto *x);	
----------	--------------------------------------	--

## 20.3 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);	
mousedrag	void scripto_ui_mousedrag(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);	

## 20.4 Message for Audio Objects

dsp	void plus_dsp(t_plus *x, t_signal **sp, short *count);	
-----	--	--

## 20.5 Messages for Objects Containing Text Fields

key	long textfield_key(t_textfield *x, t_object *patcherview, long keycode, long modifiers, long textcharacter);	
keyfilter	long textfield_keyfilter(t_ textfield *x, t_object *patcherview, long *keycode, long *modifiers, long *textcharacter);	
enter	void textfield_enter(t_textfield *x);	
select	void textfield_select(t_textfield *x);	

## 20.6 Messages for Objects with Text Editor Windows

edclose	void simpletext_edclose(t_ simpletext *x, char **text, long size);	
---------	--	--

## 20.7 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 21**

# **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.

## 21.1 Object SVG Icon

To see the icons provided by Cycling '74 for objects included in Max, look in the **Cycling '74/object-palettes** 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 `main()` function) with which they are associated. You will need to place your svg in this folder for it to be found by Max.

*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.*

## 21.2 Object Palette Definition

Adding the svg file will make the icon available to Max for use in some ways. To make your icon appear in the new object palette, however, you must create a palette containing your SVG file. If you look in the **Cycling '74/object-palettes** folder (where you placed your SVG file), you should notice some files with names like "palette1.json", "palette2.json", and "palette3.json". For your object, you should create a new palette file.

For the following example we will assume you have created an object called 'littleuifoo'. For this object we will create a palette called 'littleuifoo-palette.json'. The contents of this file will look like this:

```
{
  "patcher" : {
    "rect" : [ 0.000000, 0.000000, 1000.000000, 1000.000000 ],
    "bgcolor" : [ 1.000000, 1.000000, 1.000000, 1.000000 ],
    "bglocked" : 0,
    "defrect" : [ 10.000000, 59.000000, 1176.000000, 668.000000 ],
    "boxes" : [ {
      "box" : {
        "maxclass" : "fpic",
        "boxalpha" : 1.000000,
        "presentation" : 0,
        "destrect" : [ 0.000000, 0.000000, 0.000000, 0.000000 ],
        "patching_rect" : [ 241.000000, 244.000000, 100.000000, 50.000000 ],
        "autofit" : 0,
        "id" : "obj-1",
        "ignoreclick" : 0,
        "hidden" : 0,
        "fontname" : "Courier",
        "pic" : "littleuifoo.svg",
        "xoffset" : 0.000000,
        "yoffset" : 0.000000,
        "background" : 0,
        "presentation_rect" : [ 0.000000, 0.000000, 0.000000, 0.000000 ],
        "fontsize" : 12.000000,
        "instance_attributes" : {
          "palette_category" : [ "Images", "Interface" ],
          "palette_action" : "littleuifoo"
        }
      }
    }
  ]
}
```



```
}  
  
}  
  ],  
  "lines" : [  ]  
}  
}
```

Most of this palette file will be the same for any given object. The astute reader might notice that this is a JSON representation of a [t\\_dictionary](#) representing a patcher that includes an **fpic** object. We care about three lines in this dictionary:

1. The 'pic' attribute of the fpic object determines what image will be displayed in the new-object palette.
2. The 'palette\_category' instance attribute will determine what categories/tabs your icon will appear under in the new-object palette.
3. The 'palette\_action' instance attribute will determine what object class is instantiated by when user chooses your icon.



## **Chapter 22**

# **Max Development for the Mac OS**

## 22.1 Development Environment

- Copy frameworks (MaxAPI.framework etc.) from/Applications/-MaxMSP.app/Contents/Frameworksto/Library/Frameworks
- folder structure:

```
Max5Dev
c74support
myObject
myObject.c
myObject.xcodeproj
myOtherObject
myOtherObject.c
myOtherObject.xcodeproj
sysbuild
Development
MaxMSP Runtime.app
Cycling '74
max-externals
myObject.mxo
myOtherObject.mxo
```

## 22.2 Debugging

- Make a patch with your object and messages to test it (object can be bogus)
- Set MaxMSP Runtime to be executable for your object (this is saved locally)
- Build and Debug
- Set breakpoint in your object

## 22.3 What to check if it doesn't work...

- Info.plist?
- Is there an executable?
- Assign class variable?
- Register class?
- Return instance from new instance routine?

## **Chapter 23**

# **Max Development for Windows**

## 23.1 Development Environment

- TBD

## 23.2 Debugging

- Is it possible to debug on Windows using the Runtime?

## 23.3 What to check if it doesn't work...

- Are you running on Mac? That could solve the problem ;-)

# Chapter 24

## Module Index

### 24.1 Modules

Here is a list of all modules:

Attributes . . . . .	117
Classes . . . . .	190
Old-Style Classes . . . . .	197
Inlets and Outlets . . . . .	203
Data Storage . . . . .	210
Atom Array . . . . .	213
Database . . . . .	219
Dictionary . . . . .	234
Hash Table . . . . .	256
Index Map . . . . .	266
Linked List . . . . .	271
Quick Map . . . . .	286
String Object . . . . .	289
Symbol Object . . . . .	291
Data Types . . . . .	293
Atoms . . . . .	295
Atombufs . . . . .	314
Binbufs . . . . .	316
Symbols . . . . .	322
Files and Folders . . . . .	324
Memory Management . . . . .	347
Miscellaneous . . . . .	355
Console . . . . .	366
Byte Ordering . . . . .	371
Extending expr . . . . .	377
Table Access . . . . .	381
Text Editor Windows . . . . .	383
Presets . . . . .	384
Event and File Serial Numbers . . . . .	387
Loading Max Files . . . . .	389
Monitors and Displays . . . . .	392
Windows . . . . .	394

Mouse and Keyboard . . . . .	395
Example Projects . . . . .	399
MSP . . . . .	402
Buffers . . . . .	409
PFFT . . . . .	411
Poly . . . . .	412
Objects . . . . .	413
Patcher . . . . .	437
jpatcher . . . . .	439
jbox . . . . .	452
jpatchline . . . . .	474
jpatcherview . . . . .	478
Timing . . . . .	483
Clocks . . . . .	484
Qelems . . . . .	493
System API . . . . .	496
ITM Time Objects . . . . .	500
Threads . . . . .	514
Critical Regions . . . . .	522
Mutexes . . . . .	525
User Interface . . . . .	528
JGraphics . . . . .	529
JSurface . . . . .	545
Scalable Vector Graphics . . . . .	555
JFont . . . . .	557
JGraphics Matrix Transformations . . . . .	562
JPattern . . . . .	566
Colors . . . . .	567
TextField . . . . .	570
TextLayout . . . . .	580
Popup Menus . . . . .	584
Box Layer . . . . .	588
DataView . . . . .	591
Unicode . . . . .	593



## Chapter 25

# Data Structure Index

### 25.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">Ex_ex</a> ( <a href="#">Ex_ex</a> ) . . . . .	597
<a href="#">t_atom</a> (An atom is a typed datum) . . . . .	598
<a href="#">t_atomarray</a> (The atomarray object) . . . . .	599
<a href="#">t_atombuf</a> (The atombuf struct provides a way to pass a collection of atoms) . . . . .	600
<a href="#">t_attr</a> (Common attr struct) . . . . .	601
<a href="#">t_buffer</a> (Data structure for the buffer~ object) . . . . .	602
<a href="#">t_celldesc</a> (A dataview cell description) . . . . .	605
<a href="#">t_charset_converter</a> (The charset_converter object) . . . . .	606
<a href="#">t_class</a> (The data structure for a Max class) . . . . .	607
<a href="#">t_datetime</a> (The SysTime data structure) . . . . .	608
<a href="#">t_dictionary</a> (The dictionary object) . . . . .	609
<a href="#">t_dictionary_entry</a> (A dictionary entry) . . . . .	611
<a href="#">t_expr</a> (Struct for an instance of expr) . . . . .	612
<a href="#">t_fileinfo</a> (Information about a file) . . . . .	613
<a href="#">t_funbuff</a> (The structure of a funbuff object) . . . . .	614
<a href="#">t_hashtab</a> (The hashtab object) . . . . .	616
<a href="#">t_hashtab_entry</a> (A hashtab entry) . . . . .	617
<a href="#">t_indexmap</a> (An indexmap object) . . . . .	618
<a href="#">t_indexmap_entry</a> (An indexmap element) . . . . .	620
<a href="#">t_jbox</a> (The <a href="#">t_jbox</a> struct provides the header for a Max user-interface object) . . . . .	621
<a href="#">t_jboxdrawparams</a> (The <a href="#">t_jboxdrawparams</a> structure) . . . . .	622
<a href="#">t_jcolumn</a> (A dataview column) . . . . .	623
<a href="#">t_jdataview</a> (The dataview object) . . . . .	627
<a href="#">t_jgraphics_font_extents</a> (A structure for holding information related to how much space the rendering of a given font will use) . . . . .	631
<a href="#">t_jmatrix</a> (An affine transformation (such as scale, shear, etc)) . . . . .	632
<a href="#">t_jrgb</a> (A color composed of red, green, and blue components) . . . . .	633
<a href="#">t_jrgba</a> (A color composed of red, green, blue, and alpha components) . . . . .	634
<a href="#">t_linklist</a> (The linklist object) . . . . .	635
<a href="#">t_llelem</a> (A linklist element) . . . . .	636
<a href="#">t_messlist</a> (A list of symbols and their corresponding methods, complete with typechecking information) . . . . .	637

<a href="#">t_object</a> (The structure for the head of any object which wants to have inlets or outlets, or support attributes ) . . . . .	638
<a href="#">t_path</a> (The path data structure ) . . . . .	639
<a href="#">t_pathlink</a> (The pathlink data structure ) . . . . .	640
<a href="#">t_pfftpub</a> (Public FFT Patcher struct ) . . . . .	641
<a href="#">t_privatesortrec</a> (Used to pass data to a client sort function ) . . . . .	643
<a href="#">t_pt</a> (Coordinates for specifying a point ) . . . . .	645
<a href="#">t_pxjbox</a> (Header for any ui signal processing object ) . . . . .	646
<a href="#">t_pxobject</a> (Header for any non-ui signal processing object ) . . . . .	647
<a href="#">t_quickmap</a> (The quickmap object ) . . . . .	648
<a href="#">t_rect</a> (Coordinates for specifying a rectangular region ) . . . . .	649
<a href="#">t_signal</a> (The signal data structure ) . . . . .	650
<a href="#">t_size</a> (Coordinates for specifying the size of a region ) . . . . .	651
<a href="#">t_string</a> (The string object ) . . . . .	652
<a href="#">t_symbol</a> (The symbol ) . . . . .	653
<a href="#">t_symobject</a> (The symobject data structure ) . . . . .	654
<a href="#">t_tinyobject</a> (The tiny object structure sits at the head of any object to which you may pass messages (and which you may feed to <a href="#">freeobject()</a> ) ) . . . . .	655
<a href="#">t_zll</a> (A simple doubly-linked list used by the <a href="#">t_funbuff</a> object ) . . . . .	656
<a href="#">word</a> (Union for packing any of the datum defined in <a href="#">e_max_atomtypes</a> ) . . . . .	657

# Chapter 26

## Module Documentation

### 26.1 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.*

#### Defines

- #define [CLASS\\_ATTR\\_CHAR](#)(c, attrname, flags, structname, structmember) class\_  
addattr((c),attr\_offset\_new(attrname,USESVM(char),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))

*Create a char attribute and add it to a Max class.*

- #define [CLASS\\_ATTR\\_LONG](#)(c, attrname, flags, structname, structmember) class\_  
addattr((c),attr\_offset\_new(attrname,USESVM(long),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))

*Create a long integer attribute and add it to a Max class.*

- #define [CLASS\\_ATTR\\_FLOAT](#)(c, attrname, flags, structname, structmember) class\_  
addattr((c),attr\_offset\_new(attrname,USESVM(float32),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))

*Create a 32-bit float attribute and add it to a Max class.*

- #define [CLASS\\_ATTR\\_DOUBLE](#)(c, attrname, flags, structname, structmember) class\_  
addattr((c),attr\_offset\_new(attrname,USESVM(float64),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))

*Create a 64-bit float attribute and add it to a Max class.*

- #define [CLASS\\_ATTR\\_SYM](#)(c, attrname, flags, structname, structmember) class\_addattr((c),attr\_  
offset\_new(attrname,USESVM(symbol),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))

Create a `t_symbol*` attribute and add it to a Max class.

- #define `CLASS_ATTR_ATOM(c, attrname, flags, structname, structmember)` `class_`  
`addattr((c),attr_offset_new(attrname,USESVM(atom),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))`

Create a `t_atom` attribute and add it to a Max class.

- #define `CLASS_ATTR_OBJ(c, attrname, flags, structname, structmember)` `class_addattr((c),attr_`  
`offset_new(attrname,USESVM(object),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))`

Create a `t_object*` attribute and add it to a Max class.

- #define `CLASS_ATTR_CHAR_ARRAY(c, attrname, flags, struct-`  
`name, structmember, size) class_addattr((c),attr_offset_array_`  
`new(attrname,USESVM(char),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmember)))`

Create an array-of-chars attribute of fixed length, and add it to a Max class.

- #define `CLASS_ATTR_LONG_ARRAY(c, attrname, flags, struct-`  
`name, structmember, size) class_addattr((c),attr_offset_array_`  
`new(attrname,USESVM(long),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmember)))`

Create an array-of-long-integers attribute of fixed length, and add it to a Max class.

- #define `CLASS_ATTR_FLOAT_ARRAY(c, attrname, flags, struct-`  
`name, structmember, size) class_addattr((c),attr_offset_array_`  
`new(attrname,USESVM(float32),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmember)))`

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, struct-`  
`name, structmember, size) class_addattr((c),attr_offset_array_`  
`new(attrname,USESVM(float64),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmember)))`

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, struct-`  
`name, structmember, size) class_addattr((c),attr_offset_array_`  
`new(attrname,USESVM(symbol),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmember)))`

Create an array-of-symbols attribute of fixed length, and add it to a Max class.

- #define `CLASS_ATTR_ATOM_ARRAY(c, attrname, flags, struct-`  
`name, structmember, size) class_addattr((c),attr_offset_array_`  
`new(attrname,USESVM(atom),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmember)))`

Create an array-of-atoms attribute of fixed length, and add it to a Max class.

- #define `CLASS_ATTR_OBJ_ARRAY(c, attrname, flags, struct-`  
`name, structmember, size) class_addattr((c),attr_offset_array_`  
`new(attrname,USESVM(object),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmember)))`

*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) class_addattr((c),attr_offset_array_-new(attrname,USESVM(char),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember),calcoffset(s`

*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) class_addattr((c),attr_offset_array_-new(attrname,USESVM(long),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember),calcoffset(s`

*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) class_addattr((c),attr_offset_array_-new(attrname,USESVM(float32),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember),calcoffset(s`

*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) class_addattr((c),attr_offset_array_-new(attrname,USESVM(float64),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember),calcoffset(s`

*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) class_addattr((c),attr_offset_array_-new(attrname,USESVM(symbol),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember),calcoffset(s`

*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) class_addattr((c),attr_offset_array_-new(attrname,USESVM(atom),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember),calcoffset(s`

*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) class_addattr((c),attr_offset_array_-new(attrname,USESVM(object),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember),calcoffset(s`

*Create an array-of-objects attribute of variable length, and add it to a Max class.*

- #define `STRUCT_ATTR_CHAR(c, flags, structname, structmember) CLASS_ATTR_CHAR(c,#structmember,flags,structname,structmember)`

*Create a char attribute and add it to a Max class.*

- #define `STRUCT_ATTR_LONG(c, flags, structname, structmember) CLASS_ATTR_LONG(c,#structmember,flags,structname,structmember)`

*Create a long integer attribute and add it to a Max class.*

- #define `STRUCT_ATTR_FLOAT(c, flags, structname, structmember)` `CLASS_ATTR_FLOAT(c,#structmember,flags,structname,structmember)`  
*Create a 32bit float attribute and add it to a Max class.*
- #define `STRUCT_ATTR_DOUBLE(c, flags, structname, structmember)` `CLASS_ATTR_DOUBLE(c,#structmember,flags,structname,structmember)`  
*Create a 64bit float attribute and add it to a Max class.*
- #define `STRUCT_ATTR_SYM(c, flags, structname, structmember)` `CLASS_ATTR_SYM(c,#structmember,flags,structname,structmember)`  
*Create a `t_symbol*` attribute and add it to a Max class.*
- #define `STRUCT_ATTR_ATOM(c, flags, structname, structmember)` `CLASS_ATTR_ATOM(c,#structmember,flags,structname,structmember)`  
*Create a `t_atom` attribute and add it to a Max class.*
- #define `STRUCT_ATTR_OBJ(c, flags, structname, structmember)` `CLASS_ATTR_OBJ(c,#structmember,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)` `CLASS_ATTR_CHAR_ARRAY(c,#structmember,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)` `CLASS_ATTR_LONG_ARRAY(c,#structmember,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)` `CLASS_ATTR_FLOAT_ARRAY(c,#structmember,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)` `CLASS_ATTR_DOUBLE_ARRAY(c,#structmember,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)` `CLASS_ATTR_SYM_ARRAY(c,#structmember,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)` `CLASS_ATTR_ATOM_ARRAY(c,#structmember,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)` `CLASS_ATTR_OBJ_ARRAY(c,#structmember,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)` `CLASS_ATTR_CHAR_VARSIZE(c,#structmember,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) CLASS_ATTR_LONG_VARSIZE(c,#structmember,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) CLASS_ATTR_FLOAT_VARSIZE(c,#structmember,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) CLASS_ATTR_DOUBLE_VARSIZE(c,#structmember,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) CLASS_ATTR_SYM_VARSIZE(c,#structmember,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) CLASS_ATTR_ATOM_VARSIZE(c,#structmember,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) CLASS_ATTR_OBJ_VARSIZE(c,#structmember,flags,structname,structmember,sizemember,maxsize)`

*Create an array-of-objects attribute of variable length, and add it to a Max class.*

- #define `STATIC_ATTR_CHAR(c, attrname, flags, val) STATIC_ATTR_FORMAT(c,attrname,USES_SYM(char),flags,"c",val)`

*Create a shared (static/global) char attribute and add it to a Max class.*

- #define `STATIC_ATTR_LONG(c, attrname, flags, val) STATIC_ATTR_FORMAT(c,attrname,USES_SYM(long),flags,"l",val)`

*Create a shared (static/global) long integer attribute and add it to a Max class.*

- #define `STATIC_ATTR_FLOAT(c, attrname, flags, val) STATIC_ATTR_FORMAT(c,attrname,USES_SYM(float32),flags,"f",val)`

*Create a shared (static/global) 32bit float attribute and add it to a Max class.*

- #define `STATIC_ATTR_DOUBLE(c, attrname, flags, val) STATIC_ATTR_FORMAT(c,attrname,USES_SYM(float64),flags,"d",val)`

*Create a shared (static/global) 64bit float attribute and add it to a Max class.*

- #define `STATIC_ATTR_SYM(c, attrname, flags, val) STATIC_ATTR_FORMAT(c,attrname,USES_SYM(symbol),flags,"s",val)`

*Create a shared (static/global) `t_symbol*` attribute and add it to a Max class.*

- #define [STATIC\\_ATTR\\_ATOM](#)(c, attrname, flags, val) [STATIC\\_ATTR\\_FORMAT](#)(c,attrname,USESVM(atom),flags,"a",val)  
*Create a shared (static/global) [t\\_atom](#) attribute and add it to a Max class.*
- #define [STATIC\\_ATTR\\_OBJ](#)(c, attrname, flags, val) [STATIC\\_ATTR\\_FORMAT](#)(c,attrname,USESVM(object),flags,"o",val)  
*Create a shared (static/global) [t\\_object\\*](#) attribute and add it to a Max class.*
- #define [STATIC\\_ATTR\\_CHAR\\_ARRAY](#)(c, attrname, flags, count, vals) [STATIC\\_ATTR\\_FORMAT](#)(c,attrname,USESVM(char),flags,"C",count,vals)  
*Create a shared (static/global) array-of-chars attribute of fixed length, and add it to a Max class.*
- #define [STATIC\\_ATTR\\_LONG\\_ARRAY](#)(c, attrname, flags, count, vals) [STATIC\\_ATTR\\_FORMAT](#)(c,attrname,USESVM(long),flags,"L",count,vals)  
*Create a shared (static/global) array-of-long-integers attribute of fixed length, and add it to a Max class.*
- #define [STATIC\\_ATTR\\_FLOAT\\_ARRAY](#)(c, attrname, flags, count, vals) [STATIC\\_ATTR\\_FORMAT](#)(c,attrname,USESVM(float32),flags,"F",count,vals)  
*Create a shared (static/global) array-of-32bit-floats attribute of fixed length, and add it to a Max class.*
- #define [STATIC\\_ATTR\\_DOUBLE\\_ARRAY](#)(c, attrname, flags, count, vals) [STATIC\\_ATTR\\_FORMAT](#)(c,attrname,USESVM(float64),flags,"D",count,vals)  
*Create a shared (static/global) array-of-64bit-floats attribute of fixed length, and add it to a Max class.*
- #define [STATIC\\_ATTR\\_SYM\\_ARRAY](#)(c, attrname, flags, count, vals) [STATIC\\_ATTR\\_FORMAT](#)(c,attrname,USESVM(symbol),flags,"S",count,vals)  
*Create a shared (static/global) array-of-symbols attribute of fixed length, and add it to a Max class.*
- #define [STATIC\\_ATTR\\_ATOM\\_ARRAY](#) [STATIC\\_ATTR\\_ATOMS](#)  
*Create a shared (static/global) array-of-atoms attribute of fixed length, and add it to a Max class.*
- #define [STATIC\\_ATTR\\_OBJ\\_ARRAY](#)(c, attrname, flags, count, vals) [STATIC\\_ATTR\\_FORMAT](#)(c,attrname,USESVM(object),flags,"O",count,vals)  
*Create a shared (static/global) array-of-objects attribute of fixed length, and add it to a Max class.*
- #define [OBJ\\_ATTR\\_CHAR](#)(x, attrname, flags, val) [OBJ\\_ATTR\\_FORMAT](#)(x,attrname,USESVM(char),flags,"c",val)  
*Create an instance-local char attribute and add it to a Max class.*
- #define [OBJ\\_ATTR\\_LONG](#)(x, attrname, flags, val) [OBJ\\_ATTR\\_FORMAT](#)(x,attrname,USESVM(long),flags,"l",val)  
*Create an instance-local long integer attribute and add it to a Max class.*
- #define [OBJ\\_ATTR\\_FLOAT](#)(x, attrname, flags, val) [OBJ\\_ATTR\\_FORMAT](#)(x,attrname,USESVM(float32),flags,"f",val)  
*Create an instance-local 32bit float attribute and add it to a Max class.*
- #define [OBJ\\_ATTR\\_DOUBLE](#)(x, attrname, flags, val) [OBJ\\_ATTR\\_FORMAT](#)(x,attrname,USESVM(float64),flags,"d",val)  
*Create an instance-local 64bit float attribute and add it to a Max class.*



- #define `OBJ_ATTR_SYM(x, attrname, flags, val)` `OBJ_ATTR_-  
FORMAT(x,attrname,USESYM(symbol),flags,"s",val)`  
*Create an instance-local `t_symbol*` attribute and add it to a Max class.*
- #define `OBJ_ATTR_ATOM(x, attrname, flags, val)` `OBJ_ATTR_-  
FORMAT(x,attrname,USESYM(atom),flags,"a",val)`  
*Create an instance-local `t_atom` attribute and add it to a Max class.*
- #define `OBJ_ATTR_OBJ(x, attrname, flags, val)` `OBJ_ATTR_-  
FORMAT(x,attrname,USESYM(object),flags,"o",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)` `OBJ_ATTR_-  
FORMAT(x,attrname,USESYM(char),flags,"C",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)` `OBJ_ATTR_-  
FORMAT(x,attrname,USESYM(long),flags,"L",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)` `OBJ_ATTR_-  
FORMAT(x,attrname,USESYM(float32),flags,"F",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)` `OBJ_ATTR_-  
FORMAT(x,attrname,USESYM(float64),flags,"D",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)` `OBJ_ATTR_-  
FORMAT(x,attrname,USESYM(symbol),flags,"S",count,vals)`  
*Create an instance-local array-of-symbols attribute of fixed length, and add it to the object.*
- #define `OBJ_ATTR_ATOM_ARRAY` `OBJ_ATTR_ATOMS`  
*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)` `OBJ_ATTR_-  
FORMAT(x,attrname,USESYM(object),flags,"O",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) { [t\\_object](#) \*theattr=([t\\_object](#) \*)class\_attr\_get(c,gensym(attrname)); [CLASS\\_ATTR\\_ATTR\\_PARSE](#)(c,attrname,"default",(t\_symbol \*)object\_method(theattr,USES\_YM(gettype)),flags,parsestr); }  
*Add a new attribute to the specified attribute to specify a default value.*
- #define [CLASS\\_ATTR\\_SAVE](#)(c, attrname, flags) [CLASS\\_ATTR\\_ATTR\\_PARSE](#)(c,attrname,"save",USES\_YM(long),flags,"1")  
*Add a new attribute to the specified attribute to indicate that the specified attribute should be saved with the patcher.*
- #define [CLASS\\_ATTR\\_DEFAULT\\_SAVE](#)(c, attrname, flags, parsestr) { [CLASS\\_ATTR\\_DEFAULT](#)(c,attrname,flags,parsestr); [CLASS\\_ATTR\\_SAVE](#)(c,attrname,flags); }  
*A convenience wrapper for both [CLASS\\_ATTR\\_DEFAULT](#) and [CLASS\\_ATTR\\_SAVE](#).*
- #define [CLASS\\_ATTR\\_DEFAULTNAME](#)(c, attrname, flags, parsestr) { [t\\_object](#) \*theattr=([t\\_object](#) \*)class\_attr\_get(c,gensym(attrname)); [CLASS\\_ATTR\\_ATTR\\_PARSE](#)(c,attrname,"defaultname",(t\_symbol \*)object\_method(theattr,USES\_YM(gettype)),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) { [CLASS\\_ATTR\\_DEFAULTNAME](#)(c,attrname,flags,parsestr); [CLASS\\_ATTR\\_SAVE](#)(c,attrname,flags); }  
*A convenience wrapper for both [CLASS\\_ATTR\\_DEFAULTNAME](#) and [CLASS\\_ATTR\\_SAVE](#).*
- #define [CLASS\\_ATTR\\_MIN](#)(c, attrname, flags, parsestr) { [t\\_object](#) \*theattr=([t\\_object](#) \*)class\_attr\_get(c,gensym(attrname)); [CLASS\\_ATTR\\_ATTR\\_PARSE](#)(c,attrname,"min",(t\_symbol \*)object\_method(theattr,USES\_YM(gettype)),flags,parsestr); }  
*Add a new attribute to the specified attribute to specify a lower range.*
- #define [CLASS\\_ATTR\\_MAX](#)(c, attrname, flags, parsestr) { [t\\_object](#) \*theattr=([t\\_object](#) \*)class\_attr\_get(c,gensym(attrname)); [CLASS\\_ATTR\\_ATTR\\_PARSE](#)(c,attrname,"max",(t\_symbol \*)object\_method(theattr,USES\_YM(gettype)),flags,parsestr); }  
*Add a new attribute to the specified attribute to specify an upper range.*
- #define [CLASS\\_ATTR\\_PAINT](#)(c, attrname, flags) [CLASS\\_ATTR\\_ATTR\\_PARSE](#)(c,attrname,"paint",USES\_YM(long),flags,"1")  
*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) { [CLASS\\_ATTR\\_DEFAULT](#)(c,attrname,flags,parsestr); [CLASS\\_ATTR\\_PAINT](#)(c,attrname,flags); }

*A convenience wrapper for both `CLASS_ATTR_DEFAULT` and `CLASS_ATTR_PAINT`.*

- #define `CLASS_ATTR_DEFAULT_SAVE_PAINT`(c, attrname, flags, parsestr) { `CLASS_ATTR_DEFAULT`(c,attrname,flags,parsestr); `CLASS_ATTR_SAVE`(c,attrname,flags); `CLASS_ATTR_PAINT`(c,attrname,flags); }

*A convenience wrapper for `CLASS_ATTR_DEFAULT`, `CLASS_ATTR_SAVE`, and `CLASS_ATTR_PAINT`.*

- #define `CLASS_ATTR_DEFAULTNAME_PAINT`(c, attrname, flags, parsestr) { `CLASS_ATTR_DEFAULTNAME`(c,attrname,flags,parsestr); `CLASS_ATTR_PAINT`(c,attrname,flags); }

*A convenience wrapper for `CLASS_ATTR_DEFAULTNAME`, `CLASS_ATTR_SAVE`, and `CLASS_ATTR_PAINT`.*

- #define `CLASS_ATTR_DEFAULTNAME_SAVE_PAINT`(c, attrname, flags, parsestr) { `CLASS_ATTR_DEFAULTNAME`(c,attrname,flags,parsestr); `CLASS_ATTR_SAVE`(c,attrname,flags); `CLASS_ATTR_PAINT`(c,attrname,flags); }

*A convenience wrapper for `CLASS_ATTR_DEFAULTNAME`, `CLASS_ATTR_SAVE`, and `CLASS_ATTR_PAINT`.*

- #define `CLASS_ATTR_STYLE`(c, attrname, flags, parsestr) `CLASS_ATTR_ATTR_PARSE`(c,attrname,"style",USESYM(symbol),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) `CLASS_ATTR_ATTR_FORMAT`(c,attrname,"label",USESYM(symbol),flags,"s",gensym(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) { `CLASS_ATTR_STYLE`(c,attrname,flags,"enum"); `CLASS_ATTR_ATTR_PARSE`(c,attrname,"enumvals",USESYM(atom),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) { `CLASS_ATTR_STYLE`(c,attrname,flags,"enumindex"); `CLASS_ATTR_ATTR_PARSE`(c,attrname,"enumvals",USESYM(atom),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) `CLASS_ATTR_ATTR_PARSE`(c,attrname,"category",USESYM(symbol),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) { `CLASS_ATTR_ATTR_PARSE`(c,attrname,"style",USESYM(symbol),flags,stylestr); `CLASS_ATTR_ATTR_FORMAT`(c,attrname,"label",USESYM(symbol),flags,"s",gensym(labelstr)); }

*A convenience wrapper for `CLASS_ATTR_STYLE`, and `CLASS_ATTR_LABEL`.*

- #define `CLASS_ATTR_INVISIBLE`(c, attrname, flags) `CLASS_ATTR_ATTR_PARSE`(c,attrname,"invisible",USESYM(long),flags,"1")

*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) [CLASS\\_ATTR\\_ATTR\\_PARSE](#)(c,attrname,"order",USESYM(long),flags,parsestr)

*Add a new attribute to the specified attribute to specify a default order in which to list attributes.*

- #define [OBJ\\_ATTR\\_DEFAULT](#)(x, attrname, flags, parsestr) { [t\\_object](#) \*theattr=([t\\_object](#) \*)[object\\_attr\\_get](#)(x,gensym(attrname)); [OBJ\\_ATTR\\_ATTR\\_PARSE](#)(x,attrname,"default",(t\_symbol \*)[object\\_method](#)(theattr,USESYM(gettype)),flags,parsestr); }

*An instance-attribute version of [CLASS\\_ATTR\\_DEFAULT](#).*

- #define [OBJ\\_ATTR\\_SAVE](#)(x, attrname, flags) [OBJ\\_ATTR\\_ATTR\\_PARSE](#)(x,attrname,"save",USESYM(long),flags,"1")

*An instance-attribute version of [CLASS\\_ATTR\\_SAVE](#).*

- #define [OBJ\\_ATTR\\_DEFAULT\\_SAVE](#)(x, attrname, flags, parsestr) { [OBJ\\_ATTR\\_DEFAULT](#)(x,attrname,flags,parsestr); [OBJ\\_ATTR\\_SAVE](#)(x,attrname,flags); }

*An instance-attribute version of [CLASS\\_ATTR\\_DEFAULT\\_SAVE](#).*

- #define [CLASS\\_STICKY\\_ATTR](#)(c, name, flags, parsestr) { [t\\_object](#) \*attr = [attribute\\_new\\_parse](#)(name,NULL,flags,parsestr); [class\\_sticky](#)(c,gensym("sticky\_attr"),gensym(name),attr); }

*Create an attribute, and add it to all following attribute declarations.*

- #define [CLASS\\_STICKY\\_ATTR\\_CLEAR](#)(c, name) [class\\_sticky\\_clear](#)(c,gensym("sticky\_attr"),name?gensym(name):NULL)

*Close a [CLASS\\_STICKY\\_ATTR](#) block.*

- #define [CLASS\\_STICKY\\_METHOD](#)(c, name, flags, parsestr) { [t\\_object](#) \*attr = [attribute\\_new\\_parse](#)(name,NULL,flags,parsestr); [class\\_sticky](#)(c,gensym("sticky\_method"),gensym(name),attr); }

*Create an attribute, and add it to all following method declarations.*

- #define [CLASS\\_STICKY\\_METHOD\\_CLEAR](#)(c, name) [class\\_sticky\\_clear](#)(c,gensym("sticky\_method"),name?gensym(name):clear)

*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](#) = 0x00000000,  
[ATTR\\_GET\\_OPAQUE](#) = 0x00000001,  
[ATTR\\_SET\\_OPAQUE](#) = 0x00000002,  
[ATTR\\_GET\\_OPAQUE\\_USER](#) = 0x00000100,  
[ATTR\\_SET\\_OPAQUE\\_USER](#) = 0x00000200,  
[ATTR\\_GET\\_DEFER](#) = 0x00010000,  
[ATTR\\_GET\\_USURP](#) = 0x00020000,

```
ATTR_GET_DEFER_LOW = 0x00040000,
ATTR_GET_USURP_LOW = 0x00080000,
ATTR_SET_DEFER     = 0x01000000,
ATTR_SET_USURP     = 0x02000000,
ATTR_SET_DEFER_LOW = 0x04000000,
ATTR_SET_USURP_LOW = 0x08000000 }
```

*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 (char *name, t\_symbol *type, long flags, method mget, method mset)`  
*Create a new attribute.*

- `t_object * attr_offset_new` (char \*name, `t_symbol` \*type, long flags, `method` mget, `method` mset, long offset)  
*Create a new attribute.*
- `t_object * attr_offset_array_new` (char \*name, `t_symbol` \*type, long size, long flags, `method` mget, `method` mset, long offsetcount, long offset)  
*Create a new attribute.*
- `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, long c)  
*Sets the value of an attribute, given its parent object and name.*
- `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, 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, 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, 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, `uchar` \*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, `uchar` \*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.*
- `t_max_err object_attr_setparse` (`t_object` \*x, `t_symbol` \*s, 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, 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.*
- `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.*

### 26.1.1 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 `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.



## 26.1.2 Setting and Getting Attribute Values

By default, Max provides standard attribute accessors. These are the functions the 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.

### 26.1.2.1 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 not 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..

### 26.1.2.2 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;
}
```

## 26.1.3 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.

## 26.1.4 Define Documentation

### 26.1.4.1 #define CLASS\_ATTR\_ACCESSORS(*c*, *attrname*, *getter*, *setter*)

**Value:**

```
{ t_object *theattr=(t_object *)class_attr_get(c,gensym(attrname)); \
    object_method(theattr,gensym("setmethod"),USESVM(get),getter); \
    object_method(theattr,gensym("setmethod"),USESVM(set),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:**

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*getter* An appropriate getter method as discussed in [Setting and Getting Attribute Values](#), or NULL to use the default getter.

*setter* An appropriate setter method as discussed in [Setting and Getting Attribute Values](#), or NULL to use the default setter.

Definition at line 1038 of file ext\_obex\_util.h.

### 26.1.4.2 #define CLASS\_ATTR\_ADD\_FLAGS(*c*, *attrname*, *flags*)

**Value:**

```
{ t_object *theattr=(t_object *)class_attr_get(c,gensym(attrname)); \
    long oldflags = object_method(theattr,gensym("getflags")); \
    object_method(theattr,gensym("setflags"),oldflags|(flags)); }
```

Add flags to an attribute.

**Parameters:**

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*flags* Any flags you wish to add to this attribute, as defined in [e\\_max\\_attrflags](#).

Definition at line 1052 of file ext\_obex\_util.h.

### 26.1.4.3 #define CLASS\_ATTR\_ALIAS(*c*, *attrname*, *aliasname*)

**Value:**

```
{ t_object *thealias; \
    t_object *theattr=(t_object *)class_attr_get(c,gensym(attrname)); \
    thealias = object_clone(theattr); \
    object_method(thealias,USESVM(setname),gensym(aliasname)); \
    class_addattr(c,thealias); \
    CLASS_ATTR_ATTR_PARSE(c,aliasname,"alias",USESVM(symbol),0,attrname); }
```

Create a new attribute that is an alias of an existing attribute.

**Parameters:**

- c* The class pointer.
- attrname* The name of the actual attribute as a C-string.
- aliasname* The name of the new alias attribute.

Definition at line 1130 of file ext\_obex\_util.h.

```
26.1.4.4 #define CLASS_ATTR_ATOM(c, attrname, flags,
        structname, structmember) class_addattr((c),attr_offset_-
        new(attrname,USESYM(atom),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))
```

Create a [t\\_atom](#) attribute and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 104 of file ext\_obex\_util.h.

```
26.1.4.5 #define CLASS_ATTR_ATOM_ARRAY(c, attrname, flags,
        structname, structmember, size) class_addattr((c),attr_offset_array_-
        new(attrname,USESYM(atom),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmen
```

Create an array-of-atoms attribute of fixed length, and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- size* The number of items in the [t\\_atom](#) array.

Definition at line 210 of file ext\_obex\_util.h.

```
26.1.4.6 #define CLASS_ATTR_ATOM_VARSIZE(c, attrname, flags, structname,
        structmember, sizemember, maxsize) class_addattr((c),attr_offset_array_-
        new(attrname,USESYM(atom),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember),
```

Create an array-of-atoms attribute of variable length, and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- sizemember* The actual number of items in the [t\\_atom](#) array at any given moment.
- maxsize* The maximum number of items in the [t\\_atom](#) array, i.e. the number of members allocated for the array in the struct.

Definition at line 323 of file `ext_obex_util.h`.

#### 26.1.4.7 **#define CLASS\_ATTR\_CATEGORY(c, attrname, flags, parsestr) CLASS\_ATTR\_ATTR\_PARSE(c,attrname,"category",USESVM(symbol),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:**

- c* The class pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).
- parsestr* A C-string, which will be parsed into an array of atoms to set the initial value.

Definition at line 1427 of file `ext_obex_util.h`.

#### 26.1.4.8 **#define CLASS\_ATTR\_CHAR(c, attrname, flags, structname, structmember) class\_addattr((c),attr\_offset\_new(attrname,USESVM(char),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))**

Create a char attribute and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 34 of file `ext_obex_util.h`.

```
26.1.4.9 #define CLASS_ATTR_CHAR_ARRAY(c, attrname, flags,
        structname, structmember, size) class_addattr((c),attr_offset_array_-
        new(attrname,USESYM(char),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmem
```

Create an array-of-chars attribute of fixed length, and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- size* The number of chars in the array.

Definition at line 135 of file ext\_obex\_util.h.

```
26.1.4.10 #define CLASS_ATTR_CHAR_VARSIZE(c, attrname, flags, structname,
        structmember, sizemember, maxsize) class_addattr((c),attr_offset_array_-
        new(attrname,USESYM(char),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember)
```

Create an array-of-chars attribute of variable length, and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- sizemember* The actual number of items in the char array at any given moment.
- maxsize* The maximum number of items in the char array, i.e. the number of members allocated for the array in the struct.

Definition at line 243 of file ext\_obex\_util.h.

```
26.1.4.11 #define CLASS_ATTR_DEFAULT(c, attrname, flags, parsestr) {
        t_object *theattr=(t_object *)class_attr_get(c,gensym(attrname));
        CLASS_ATTR_ATTR_PARSE(c,attrname,"default",(t_symbol
        *)object_method(theattr,USESYM(gettype)),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:**

- c* The class pointer.

*attrname* The name of the attribute as a C-string.

*flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).

*parsestr* A C-string, which will be parsed into an array of atoms to set the initial value.

Definition at line 1158 of file ext\_obex\_util.h.

```
26.1.4.12 #define CLASS_ATTR_DEFAULT_PAINT(c, attrname, flags, parsestr) { CLASS_
ATTR_DEFAULT(c,attrname,flags,parsestr); CLASS_ATTR_PAINT(c,attrname,flags);
}
```

A convenience wrapper for both [CLASS\\_ATTR\\_DEFAULT](#) and [CLASS\\_ATTR\\_PAINT](#).

#### Parameters:

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).

*parsestr* 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](#)

Definition at line 1281 of file ext\_obex\_util.h.

```
26.1.4.13 #define CLASS_ATTR_DEFAULT_SAVE(c, attrname, flags, parsestr) { CLASS_
ATTR_DEFAULT(c,attrname,flags,parsestr); CLASS_ATTR_SAVE(c,attrname,flags);
}
```

A convenience wrapper for both [CLASS\\_ATTR\\_DEFAULT](#) and [CLASS\\_ATTR\\_SAVE](#).

#### Parameters:

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).

*parsestr* 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](#)

Definition at line 1185 of file ext\_obex\_util.h.

```
26.1.4.14 #define CLASS_ATTR_DEFAULT_SAVE_PAINT(c, attrname, flags,
parsestr) { CLASS_ATTR_DEFAULT(c,attrname,flags,parsestr);
CLASS_ATTR_SAVE(c,attrname,flags); CLASS_ATTR_PAINT(c,attrname,flags); }
```

A convenience wrapper for [CLASS\\_ATTR\\_DEFAULT](#), [CLASS\\_ATTR\\_SAVE](#), and [CLASS\\_ATTR\\_PAINT](#).

**Parameters:**

- c* The class pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).
- parsestr* 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](#)

Definition at line 1297 of file ext\_obex\_util.h.

```
26.1.4.15 #define CLASS_ATTR_DEFAULTNAME(c, attrname, flags, parsestr) {
    t_object *theattr=(t_object *)class_attr_get(c,gensym(attrname));
    CLASS_ATTR_ATTR_PARSE(c,attrname,"defaultname",(t_symbol
    *)object_method(theattr,USESYM(gettype)),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:**

- c* The class pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).
- parsestr* A C-string, which will be parsed into an array of atoms to set the initial value.

Definition at line 1202 of file ext\_obex\_util.h.

```
26.1.4.16 #define CLASS_ATTR_DEFAULTNAME_PAINT(c, attrname, flags,
    parsestr) { CLASS_ATTR_DEFAULTNAME(c,attrname,flags,parsestr);
    CLASS_ATTR_PAINT(c,attrname,flags); }
```

A convenience wrapper for [CLASS\\_ATTR\\_DEFAULTNAME](#), [CLASS\\_ATTR\\_SAVE](#), and [CLASS\\_ATTR\\_PAINT](#).

**Parameters:**

- c* The class pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).
- parsestr* 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](#)

Definition at line 1313 of file ext\_obex\_util.h.

```
26.1.4.17 #define CLASS_ATTR_DEFAULTNAME_SAVE(c, attrname, flags,
        parsestr) { CLASS_ATTR_DEFAULTNAME(c,attrname,flags,parsestr);
        CLASS_ATTR_SAVE(c,attrname,flags); }
```

A convenience wrapper for both [CLASS\\_ATTR\\_DEFAULTNAME](#) and [CLASS\\_ATTR\\_SAVE](#).

**Parameters:**

- c* The class pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).
- parsestr* 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](#)

Definition at line 1217 of file ext\_obex\_util.h.

```
26.1.4.18 #define CLASS_ATTR_DEFAULTNAME_SAVE_PAINT(c, attrname, flags,
        parsestr) { CLASS_ATTR_DEFAULTNAME(c,attrname,flags,parsestr);
        CLASS_ATTR_SAVE(c,attrname,flags); CLASS_ATTR_PAINT(c,attrname,flags); }
```

A convenience wrapper for [CLASS\\_ATTR\\_DEFAULTNAME](#), [CLASS\\_ATTR\\_SAVE](#), and [CLASS\\_ATTR\\_PAINT](#).

**Parameters:**

- c* The class pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).
- parsestr* 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](#)

Definition at line 1329 of file ext\_obex\_util.h.

```
26.1.4.19 #define CLASS_ATTR_DOUBLE(c, attrname, flags,
        structname, structmember) class_addattr((c),attr_offset_
        new(attrname,USESYM(float64),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))
```

Create a 64-bit float attribute and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.



*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 76 of file ext\_obex\_util.h.

```
26.1.4.20 #define CLASS_ATTR_DOUBLE_ARRAY(c, attrname, flags,
        structname, structmember, size) class_addattr((c),attr_offset_array_-
        new(attrname,USESVM(float64),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmember))
```

Create an array-of-64bit-floats attribute of fixed length, and add it to a Max class.

#### Parameters:

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.

*size* The number of doubles in the array.

Definition at line 180 of file ext\_obex\_util.h.

```
26.1.4.21 #define CLASS_ATTR_DOUBLE_VARSIZE(c, attrname, flags, structname,
        structmember, sizemember, maxsize) class_addattr((c),attr_offset_array_-
        new(attrname,USESVM(float64),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember))
```

Create an array-of-64bit-floats attribute of variable length, and add it to a Max class.

#### Parameters:

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.

*sizemember* The actual number of items in the double array at any given moment.

*maxsize* The maximum number of items in the double array, i.e. the number of members allocated for the array in the struct.

Definition at line 291 of file ext\_obex\_util.h.

```
26.1.4.22 #define CLASS_ATTR_ENUM(c, attrname, flags, parsestr) {
          CLASS_ATTR_STYLE(c,attrname,flags,"enum"); CLASS_ATTR_-
          ATTR_PARSE(c,attrname,"enumvals",USESYM(atom),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:**

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).

*parsestr* 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](#)

Definition at line 1390 of file ext\_obex\_util.h.

```
26.1.4.23 #define CLASS_ATTR_ENUMINDEX(c, attrname, flags, parsestr) {
          CLASS_ATTR_STYLE(c,attrname,flags,"enumindex"); CLASS_ATTR_-
          ATTR_PARSE(c,attrname,"enumvals",USESYM(atom),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:**

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).

*parsestr* 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](#)

Definition at line 1411 of file ext\_obex\_util.h.

**26.1.4.24 #define CLASS\_ATTR\_FILTER\_CLIP(c, attrname, minval, maxval)****Value:**

```
{ t_object *theattr=(t_object *)class_attr_get(c,gensym(attrname)); \
    attr_addfilter_clip(theattr,minval,maxval,1,1); }
```

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:**

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*minval* The maximum acceptable value to which the attribute will be limited.

*maxval* The maximum acceptable value to which the attribute will be limited.

**See also:**

Definition at line 1117 of file ext\_obex\_util.h.

**26.1.4.25 #define CLASS\_ATTR\_FILTER\_MAX(c, attrname, maxval)****Value:**

```
{ t_object *theattr=(t_object *)class_attr_get(c,gensym(attrname)); \
    attr_addfilter_clip(theattr,0,maxval,0,1); }
```

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:**

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*maxval* The maximum acceptable value to which the attribute will be limited.

**See also:**

[CLASS\\_ATTR\\_FILTER\\_MIN](#)  
[CLASS\\_ATTR\\_FILTER\\_CLIP](#)  
[CLASS\\_ATTR\\_MAX](#)

Definition at line 1101 of file ext\_obex\_util.h.

**26.1.4.26 #define CLASS\_ATTR\_FILTER\_MIN(c, attrname, minval)****Value:**

```
{ t_object *theattr=(t_object *)class_attr_get(c,gensym(attrname)); \
    attr_addfilter_clip(theattr,minval,0,1,0); }
```

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:**

- c* The class pointer.
- attrname* The name of the attribute as a C-string.
- minval* The minimum acceptable value to which the attribute will be limited.

**See also:**

[CLASS\\_ATTR\\_FILTER\\_MAX](#)  
[CLASS\\_ATTR\\_FILTER\\_CLIP](#)  
[CLASS\\_ATTR\\_MIN](#)

Definition at line 1084 of file ext\_obex\_util.h.

```
26.1.4.27 #define CLASS_ATTR_FLOAT(c, attrname, flags,
        structname, structmember) class_addattr((c),attr_offset_-
        new(attrname,USESVM(float32),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))
```

Create a 32-bit float attribute and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 62 of file ext\_obex\_util.h.

```
26.1.4.28 #define CLASS_ATTR_FLOAT_ARRAY(c, attrname, flags,
        structname, structmember, size) class_addattr((c),attr_offset_array_-
        new(attrname,USESVM(float32),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmember)))
```

Create an array-of-32bit-floats attribute of fixed length, and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- size* The number of floats in the array.

Definition at line 165 of file ext\_obex\_util.h.

**26.1.4.29** `#define CLASS_ATTR_FLOAT_VARSIZE(c, attrname, flags, structname, structmember, sizemember, maxsize) class_addattr((c),attr_offset_array_-new(attrname,USESYM(float32),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember`

Create an array-of-32bit-floats attribute of variable length, and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.

*sizemember* The actual number of items in the float array at any given moment.

*maxsize* The maximum number of items in the float array, i.e. the number of members allocated for the array in the struct.

Definition at line 275 of file `ext_obex_util.h`.

**26.1.4.30** `#define CLASS_ATTR_INVISIBLE(c, attrname, flags) CLASS_ATTR_ATTR_PARSE(c,attrname,"invisible",USESYM(long),flags,"1")`

Add a new attribute to the specified attribute to flag an attribute as invisible to the Max inspector.

**Parameters:**

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).

Definition at line 1457 of file `ext_obex_util.h`.

**26.1.4.31** `#define CLASS_ATTR_LABEL(c, attrname, flags, labelstr) CLASS_ATTR_ATTR_FORMAT(c,attrname,"label",USESYM(symbol),flags,"s",gensym(labelstr))`

Add a new attribute to the specified attribute to specify an a human-friendly label for the Max inspector.

**Parameters:**

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).

*labelstr* A C-string, which will be parsed into an array of atoms to set the initial value.

Definition at line 1369 of file `ext_obex_util.h`.

```
26.1.4.32 #define CLASS_ATTR_LONG(c, attrname, flags,
        structname, structmember) class_addattr((c),attr_offset_-
        new(attrname,USESVM(long),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))
```

Create a long integer attribute and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 48 of file `ext_obex_util.h`.

```
26.1.4.33 #define CLASS_ATTR_LONG_ARRAY(c, attrname, flags,
        structname, structmember, size) class_addattr((c),attr_offset_array_-
        new(attrname,USESVM(long),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmember)))
```

Create an array-of-long-integers attribute of fixed length, and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.

*size* The number of longs in the array.

Definition at line 150 of file `ext_obex_util.h`.

```
26.1.4.34 #define CLASS_ATTR_LONG_VARSIZE(c, attrname, flags, structname,
        structmember, sizemember, maxsize) class_addattr((c),attr_offset_array_-
        new(attrname,USESVM(long),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember)))
```

Create an array-of-long-integers attribute of variable length, and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.

*sizemember* The actual number of items in the long array at any given moment.

*maxsize* The maximum number of items in the long array, i.e. the number of members allocated for the array in the struct.

Definition at line 259 of file ext\_obex\_util.h.

```
26.1.4.35 #define CLASS_ATTR_MAX(c, attrname, flags, parsestr) {
          t_object *theattr=(t_object *)class_attr_get(c,gensym(attrname));
          CLASS_ATTR_ATTR_PARSE(c,attrname,"max",(t_symbol
          *)object_method(theattr,USESVM(gettype)),flags,parsestr); }
```

Add a new attribute to the specified attribute to specify an upper range. The values will not be automatically limited.

#### Parameters:

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).

*parsestr* 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](#)

Definition at line 1251 of file ext\_obex\_util.h.

```
26.1.4.36 #define CLASS_ATTR_MIN(c, attrname, flags, parsestr) {
          t_object *theattr=(t_object *)class_attr_get(c,gensym(attrname));
          CLASS_ATTR_ATTR_PARSE(c,attrname,"min",(t_symbol
          *)object_method(theattr,USESVM(gettype)),flags,parsestr); }
```

Add a new attribute to the specified attribute to specify a lower range. The values will not be automatically limited.

#### Parameters:

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).

*parsestr* 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](#)

Definition at line 1234 of file ext\_obex\_util.h.

```
26.1.4.37 #define CLASS_ATTR_OBJ(c, attrname, flags,
        structname, structmember) class_addattr((c),attr_offset_-
        new(attrname,USESVM(object),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))
```

Create a `t_object*` attribute and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in `e_max_attrflags`.

*structname* The C identifier for the struct (containing a valid `t_object` header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 118 of file `ext_obex_util.h`.

```
26.1.4.38 #define CLASS_ATTR_OBJ_ARRAY(c, attrname, flags,
        structname, structmember, size) class_addattr((c),attr_offset_array_-
        new(attrname,USESVM(object),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmember)))
```

Create an array-of-objects attribute of fixed length, and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in `e_max_attrflags`.

*structname* The C identifier for the struct (containing a valid `t_object` header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.

*size* The number of items in the `t_object*` array.

Definition at line 225 of file `ext_obex_util.h`.

```
26.1.4.39 #define CLASS_ATTR_OBJ_VARSIZE(c, attrname, flags, structname,
        structmember, sizemember, maxsize) class_addattr((c),attr_offset_array_-
        new(attrname,USESVM(object),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember)))
```

Create an array-of-objects attribute of variable length, and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in `e_max_attrflags`.

*structname* The C identifier for the struct (containing a valid `t_object` header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.



*sizemember* The actual number of items in the `t_object*` array at any given moment.

*maxsize* The maximum number of items in the `t_object*` array, i.e. the number of members allocated for the array in the struct.

Definition at line 339 of file `ext_obex_util.h`.

#### 26.1.4.40 `#define CLASS_ATTR_ORDER(c, attrname, flags, parsestr) CLASS_ATTR_ATTR_PARSE(c,attrname,"order",USESVM(long),flags,parsestr)`

Add a new attribute to the specified attribute to specify a default order in which to list attributes.

##### Parameters:

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*flags* Any flags you wish to declare for this new attribute, as defined in `e_max_attrflags`.

*parsestr* 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");
```

Definition at line 1477 of file `ext_obex_util.h`.

#### 26.1.4.41 `#define CLASS_ATTR_PAINT(c, attrname, flags) CLASS_ATTR_ATTR_PARSE(c,attrname,"paint",USESVM(long),flags,"1")`

Add a new attribute indicating that any changes to the specified attribute will trigger a call to the object's paint method.

##### Parameters:

*c* The class pointer.

*attrname* The name of the attribute as a C-string.

*flags* Any flags you wish to declare for this new attribute, as defined in `e_max_attrflags`.

Definition at line 1266 of file `ext_obex_util.h`.

#### 26.1.4.42 `#define CLASS_ATTR_REMOVE_FLAGS(c, attrname, flags)`

##### Value:

```
{ t_object *theattr=(t_object *)class_attr_get(c,gensym(attrname)); \
    long oldflags = object_method(theattr,gensym("getflags")); \
    object_method(theattr,gensym("setflags"),oldflags&~(flags)); }
```

Remove flags from an attribute.

**Parameters:**

- c* The class pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to remove from this attribute, as defined in [e\\_max\\_attrflags](#).

Definition at line 1066 of file ext\_obex\_util.h.

**26.1.4.43 #define CLASS\_ATTR\_RGBA(c, attrname, flags, structname, structmember)****Value:**

```
{ CLASS_ATTR_DOUBLE_ARRAY(c, attrname, flags, structname, structmember, 4); \
  CLASS_ATTR_ACCESSORS(c, attrname, NULL, jgraphics_attr_setrgba); \
  CLASS_ATTR_PAINT(c, attrname, 0); }
```

Create a color ([t\\_jrgba](#)) attribute and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 1505 of file jgraphics.h.

**26.1.4.44 #define CLASS\_ATTR\_SAVE(c, attrname, flags) CLASS\_ATTR\_ATTR\_PARSE(c,attrname,"save",USESYM(long),flags,"1")**

Add a new attribute to the specified attribute to indicate that the specified attribute should be saved with the patcher.

**Parameters:**

- c* The class pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).

Definition at line 1170 of file ext\_obex\_util.h.

**26.1.4.45 #define CLASS\_ATTR\_STYLE(c, attrname, flags, parsestr) CLASS\_ATTR\_ATTR\_PARSE(c,attrname,"style",USESYM(symbol),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:**

- c* The class pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).
- parsestr* A C-string, which will be parsed into an array of atoms to set the initial value.

Definition at line 1356 of file [ext\\_obex\\_util.h](#).

```
26.1.4.46 #define CLASS_ATTR_STYLE_LABEL(c, attrname,
        flags, stylestr, labelstr) { CLASS_ATTR_ATTR_-
        PARSE(c,attrname,"style",USESVM(symbol),flags,stylestr); CLASS_ATTR_-
        ATTR_FORMAT(c,attrname,"label",USESVM(symbol),flags,"s",gensym(labelstr));
        }
```

A convenience wrapper for [CLASS\\_ATTR\\_STYLE](#), and [CLASS\\_ATTR\\_LABEL](#).

**Parameters:**

- c* The class pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).
- stylestr* A C-string that names the style for the attribute. See [CLASS\\_ATTR\\_STYLE](#) for the available styles.
- labelstr* A C-string that names the category to which the attribute is assigned in the inspector.

**See also:**

[CLASS\\_ATTR\\_STYLE](#)  
[CLASS\\_ATTR\\_LABEL](#)

Definition at line 1445 of file [ext\\_obex\\_util.h](#).

```
26.1.4.47 #define CLASS_ATTR_SYM(c, attrname, flags,
        structname, structmember) class_addattr((c),attr_offset_-
        new(attrname,USESVM(symbol),(flags),(method)0L,(method)0L,calcoffset(structname,structmember)))
```

Create a [t\\_symbol\\*](#) attribute and add it to a Max class.

**Parameters:**

- c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 90 of file `ext_obex_util.h`.

```
26.1.4.48 #define CLASS_ATTR_SYM_ARRAY(c, attrname, flags,
        structname, structmember, size) class_addattr((c),attr_offset_array_-
        new(attrname,USESYM(symbol),(size),(flags),(method)0L,(method)0L,0/*fix*/,calcoffset(structname,structmember))
```

Create an array-of-symbols attribute of fixed length, and add it to a Max class.

#### Parameters:

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.

*size* The number of items in the [t\\_symbol\\*](#) array.

Definition at line 195 of file `ext_obex_util.h`.

```
26.1.4.49 #define CLASS_ATTR_SYM_VARSIZE(c, attrname, flags, structname,
        structmember, sizemember, maxsize) class_addattr((c),attr_offset_array_-
        new(attrname,USESYM(symbol),(maxsize),(flags),(method)0L,(method)0L,calcoffset(structname,sizemember))
```

Create an array-of-symbols attribute of variable length, and add it to a Max class.

#### Parameters:

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.

*structmember* The C identifier of the member in the struct that holds the value of this attribute.

*sizemember* The actual number of items in the [t\\_symbol\\*](#) array at any given moment.

*maxsize* The maximum number of items in the [t\\_symbol\\*](#) array, i.e. the number of members allocated for the array in the struct.

Definition at line 307 of file `ext_obex_util.h`.

```
26.1.4.50 #define CLASS_STICKY_ATTR(c, name, flags, parsestr) {
           t_object *attr = attribute_new_parse(name,NULL,flags,parsestr);
           class_sticky(c,gensym("sticky_attr"),gensym(name),attr); }
```

Create an attribute, and add it to all following attribute declarations. The block is closed by a call to [CLASS\\_STICKY\\_ATTR\\_CLEAR](#).

**Parameters:**

*c* The class pointer.

*name* The name of the new attribute to create as a C-string.

*flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).

*parsestr* 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](#)

Definition at line 1577 of file ext\_obex\_util.h.

```
26.1.4.51 #define CLASS_STICKY_ATTR_CLEAR(c, name) class_sticky_-
           clear(c,gensym("sticky_attr"),name?gensym(name):NULL)
```

Close a [CLASS\\_STICKY\\_ATTR](#) block.

**Parameters:**

*c* The class pointer.

*name* The name of the sticky attribute as a C-string.

**See also:**

[CLASS\\_STICKY\\_ATTR](#)

Definition at line 1589 of file ext\_obex\_util.h.

```
26.1.4.52 #define CLASS_STICKY_METHOD(c, name, flags, parsestr) {
    t_object *attr = attribute_new_parse(name,NULL,flags,parsestr);
    class_sticky(c,gensym("sticky_method"),gensym(name),attr); }
```

Create an attribute, and add it to all following method declarations. The block is closed by a call to [CLASS\\_STICKY\\_METHOD\\_CLEAR](#).

**Parameters:**

- c* The class pointer.
- name* The name of the new attribute to create as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).
- parsestr* 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, "undocumented", 0, "1");

// add some methods here with class_add_method()
// the undocumented attribute for methods means that the ref-page
// generator will ignore these methods.

CLASS_STICKY_ATTR_CLEAR(c, "undocumented");
```

**See also:**

[CLASS\\_STICKY\\_METHOD\\_CLEAR](#)

Definition at line 1616 of file ext\_obex\_util.h.

```
26.1.4.53 #define CLASS_STICKY_METHOD_CLEAR(c, name) class_sticky_-
    clear(c,gensym("sticky_method"),name?gensym(name):clear)
```

Close a [CLASS\\_STICKY\\_METHOD](#) block.

**Parameters:**

- c* The class pointer.
- name* The name of the sticky attribute as a C-string.

**See also:**

[CLASS\\_STICKY\\_METHOD](#)

Definition at line 1628 of file ext\_obex\_util.h.

```
26.1.4.54 #define OBJ_ATTR_ATOM(x, attrname, flags, val) OBJ_ATTR_-
    FORMAT(x,attrname,USES_SYM(atom),flags,"a",val)
```

Create an instance-local [t\\_atom](#) attribute and add it to a Max class.

**Parameters:**

- x* The object pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*val* Pointer to the value.

Definition at line 913 of file ext\_obex\_util.h.

#### 26.1.4.55 `#define OBJ_ATTR_ATOM_ARRAY OBJ_ATTR_ATOMS`

Create an instance-local array-of-atoms attribute of fixed length, and add it to the object.

##### Parameters:

*x* The object pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*count* The number of items in the [t\\_atom](#) array.

*vals* Pointer to the values.

Definition at line 1005 of file ext\_obex\_util.h.

#### 26.1.4.56 `#define OBJ_ATTR_CHAR(x, attrname, flags, val) OBJ_ATTR_FORMAT(x,attrname,USES_SYM(char),flags,"c",val)`

Create an instance-local char attribute and add it to a Max class.

##### Parameters:

*x* The object pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*val* Pointer to the value.

Definition at line 853 of file ext\_obex\_util.h.

#### 26.1.4.57 `#define OBJ_ATTR_CHAR_ARRAY(x, attrname, flags, count, vals) OBJ_ATTR_FORMAT(x,attrname,USES_SYM(char),flags,"C",count,vals)`

Create an instance-local array-of-chars attribute of fixed length, and add it to the object.

##### Parameters:

*x* The object pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*count* The number of items in the char array.

*vals* Pointer to the values.

Definition at line 940 of file ext\_obex\_util.h.

```
26.1.4.58 #define OBJ_ATTR_DEFAULT(x, attrname, flags, parsestr) {
    t_object *theattr=(t_object *)object_attr_get(x,gensym(attrname));
    OBJ_ATTR_ATTR_PARSE(x,attrname,"default",(t_symbol
    *)object_method(theattr,USESVM(gettype)),flags,parsestr); }
```

An instance-attribute version of [CLASS\\_ATTR\\_DEFAULT](#).

**Parameters:**

- x* The [t\\_object](#) instance pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).
- parsestr* A C-string, which will be parsed into an array of atoms to set the initial value.

**See also:**

[CLASS\\_ATTR\\_DEFAULT](#)

Definition at line 1515 of file ext\_obex\_util.h.

```
26.1.4.59 #define OBJ_ATTR_DEFAULT_SAVE(x, attrname, flags, parsestr) { OBJ_
    ATTR_DEFAULT(x,attrname,flags,parsestr); OBJ_ATTR_SAVE(x,attrname,flags);
    }
```

An instance-attribute version of [CLASS\\_ATTR\\_DEFAULT\\_SAVE](#).

**Parameters:**

- x* The [t\\_object](#) instance pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).
- parsestr* A C-string, which will be parsed into an array of atoms to set the initial value.

**See also:**

[CLASS\\_ATTR\\_DEFAULT\\_SAVE](#)

Definition at line 1542 of file ext\_obex\_util.h.

```
26.1.4.60 #define OBJ_ATTR_DOUBLE(x, attrname, flags, val) OBJ_ATTR_
    FORMAT(x,attrname,USESVM(float64),flags,"d",val)
```

Create an instance-local 64bit float attribute and add it to a Max class.

**Parameters:**

- x* The object pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- val* Pointer to the value.

Definition at line 889 of file ext\_obex\_util.h.



**26.1.4.61** `#define OBJ_ATTR_DOUBLE_ARRAY(x, attrname, flags, count, vals) OBJ_ATTR_FORMAT(x,attrname,USESVM(float64),flags,"D",count,vals)`

Create an instance-local array-of-64bit-floats attribute of fixed length, and add it to the object.

**Parameters:**

- x* The object pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- count* The number of items in the double array.
- vals* Pointer to the values.

Definition at line 979 of file ext\_obex\_util.h.

**26.1.4.62** `#define OBJ_ATTR_FLOAT(x, attrname, flags, val) OBJ_ATTR_FORMAT(x,attrname,USESVM(float32),flags,"f",val)`

Create an instance-local 32bit float attribute and add it to a Max class.

**Parameters:**

- x* The object pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- val* Pointer to the value.

Definition at line 877 of file ext\_obex\_util.h.

**26.1.4.63** `#define OBJ_ATTR_FLOAT_ARRAY(x, attrname, flags, count, vals) OBJ_ATTR_FORMAT(x,attrname,USESVM(float32),flags,"F",count,vals)`

Create an instance-local array-of-32bit-floats attribute of fixed length, and add it to the object.

**Parameters:**

- x* The object pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- count* The number of items in the float array.
- vals* Pointer to the values.

Definition at line 966 of file ext\_obex\_util.h.

**26.1.4.64** `#define OBJ_ATTR_LONG(x, attrname, flags, val) OBJ_ATTR_FORMAT(x,attrname,USESVM(long),flags,"I",val)`

Create an instance-local long integer attribute and add it to a Max class.

**Parameters:**

- x* The object pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- val* Pointer to the value.

Definition at line 865 of file ext\_obex\_util.h.

**26.1.4.65** `#define OBJ_ATTR_LONG_ARRAY(x, attrname, flags, count, vals) OBJ_ATTR_FORMAT(x,attrname,USESVM(long),flags,"L",count,vals)`

Create an instance-local array-of-long-integers attribute of fixed length, and add it to the object.

**Parameters:**

- x* The object pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- count* The number of items in the long array.
- vals* Pointer to the values.

Definition at line 953 of file ext\_obex\_util.h.

**26.1.4.66** `#define OBJ_ATTR_OBJ(x, attrname, flags, val) OBJ_ATTR_FORMAT(x,attrname,USESVM(object),flags,"o",val)`

Create an instance-local [t\\_object\\*](#) attribute and add it to a Max class.

**Parameters:**

- x* The object pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- val* Pointer to the value.

Definition at line 925 of file ext\_obex\_util.h.

**26.1.4.67** `#define OBJ_ATTR_OBJ_ARRAY(x, attrname, flags, count, vals) OBJ_ATTR_FORMAT(x,attrname,USESVM(object),flags,"O",count,vals)`

Create an instance-local array-of-objects attribute of fixed length, and add it to the object.

**Parameters:**

- x* The object pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- count* The number of items in the [t\\_object\\*](#) array.
- vals* Pointer to the values.

Definition at line 1018 of file ext\_obex\_util.h.

**26.1.4.68** `#define OBJ_ATTR_SAVE(x, attrname, flags) OBJ_ATTR_ATTR_-  
PARSE(x,attrname,"save",USESYM(long),flags,"1")`

An instance-attribute version of [CLASS\\_ATTR\\_SAVE](#).

**Parameters:**

- x* The [t\\_object](#) instance pointer.
- attrname* The name of the attribute as a C-string.
- flags* Any flags you wish to declare for this new attribute, as defined in [e\\_max\\_attrflags](#).

**See also:**

[CLASS\\_ATTR\\_SAVE](#)

Definition at line 1528 of file `ext_obex_util.h`.

**26.1.4.69** `#define OBJ_ATTR_SYM(x, attrname, flags, val) OBJ_ATTR_-  
FORMAT(x,attrname,USESYM(symbol),flags,"s",val)`

Create an instance-local [t\\_symbol\\*](#) attribute and add it to a Max class.

**Parameters:**

- x* The object pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- val* Pointer to the value.

Definition at line 901 of file `ext_obex_util.h`.

**26.1.4.70** `#define OBJ_ATTR_SYM_ARRAY(x, attrname, flags, count,  
vals) OBJ_ATTR_FORMAT(x,attrname,USESYM(symbol),flags,"S",count,vals)`

Create an instance-local array-of-symbols attribute of fixed length, and add it to the object.

**Parameters:**

- x* The object pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- count* The number of items in the [t\\_symbol\\*](#) array.
- vals* Pointer to the values.

Definition at line 992 of file `ext_obex_util.h`.

**26.1.4.71** `#define STATIC_ATTR_ATOM(c, attrname, flags, val) STATIC_ATTR_-  
FORMAT(c,attrname,USESYM(atom),flags,"a",val)`

Create a shared (static/global) [t\\_atom](#) attribute and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- val* Pointer to the value.

Definition at line 728 of file ext\_obex\_util.h.

**26.1.4.72 #define STATIC\_ATTR\_ATOM\_ARRAY STATIC\_ATTR\_ATOMS**

Create a shared (static/global) array-of-atoms attribute of fixed length, and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- count* The number of items in the [t\\_atom](#) array.
- vals* Pointer to the values.

Definition at line 820 of file ext\_obex\_util.h.

**26.1.4.73 #define STATIC\_ATTR\_CHAR(c, attrname, flags, val) STATIC\_ATTR\_FORMAT(c,attrname,USESVM(char),flags,"c",val)**

Create a shared (static/global) char attribute and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- val* Pointer to the value.

Definition at line 668 of file ext\_obex\_util.h.

**26.1.4.74 #define STATIC\_ATTR\_CHAR\_ARRAY(c, attrname, flags, count, vals) STATIC\_ATTR\_FORMAT(c,attrname,USESVM(char),flags,"C",count,vals)**

Create a shared (static/global) array-of-chars attribute of fixed length, and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- count* The number of items in the char array.
- vals* Pointer to the values.

Definition at line 755 of file ext\_obex\_util.h.

**26.1.4.75** `#define STATIC_ATTR_DOUBLE(c, attrname, flags, val) STATIC_ATTR_FORMAT(c,attrname,USESVM(float64),flags,"d",val)`

Create a shared (static/global) 64bit float attribute and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*val* Pointer to the value.

Definition at line 704 of file ext\_obex\_util.h.

**26.1.4.76** `#define STATIC_ATTR_DOUBLE_ARRAY(c, attrname, flags, count, vals) STATIC_ATTR_FORMAT(c,attrname,USESVM(float64),flags,"D",count,vals)`

Create a shared (static/global) array-of-64bit-floats attribute of fixed length, and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*count* The number of items in the double array.

*vals* Pointer to the values.

Definition at line 794 of file ext\_obex\_util.h.

**26.1.4.77** `#define STATIC_ATTR_FLOAT(c, attrname, flags, val) STATIC_ATTR_FORMAT(c,attrname,USESVM(float32),flags,"f",val)`

Create a shared (static/global) 32bit float attribute and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*val* Pointer to the value.

Definition at line 692 of file ext\_obex\_util.h.

**26.1.4.78** `#define STATIC_ATTR_FLOAT_ARRAY(c, attrname, flags, count, vals) STATIC_ATTR_FORMAT(c,attrname,USESVM(float32),flags,"F",count,vals)`

Create a shared (static/global) array-of-32bit-floats attribute of fixed length, and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*count* The number of items in the float array.

*vals* Pointer to the values.

Definition at line 781 of file ext\_obex\_util.h.

**26.1.4.79** `#define STATIC_ATTR_LONG(c, attrname, flags, val) STATIC_ATTR_FORMAT(c,attrname,USESVM(long),flags,"I",val)`

Create a shared (static/global) long integer attribute and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*val* Pointer to the value.

Definition at line 680 of file ext\_obex\_util.h.

**26.1.4.80** `#define STATIC_ATTR_LONG_ARRAY(c, attrname, flags, count, vals) STATIC_ATTR_FORMAT(c,attrname,USESVM(long),flags,"L",count,vals)`

Create a shared (static/global) array-of-long-integers attribute of fixed length, and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*count* The number of items in the long array.

*vals* Pointer to the values.

Definition at line 768 of file ext\_obex\_util.h.

**26.1.4.81** `#define STATIC_ATTR_OBJ(c, attrname, flags, val) STATIC_ATTR_FORMAT(c,attrname,USESVM(object),flags,"o",val)`

Create a shared (static/global) [t\\_object\\*](#) attribute and add it to a Max class.

**Parameters:**

*c* The class pointer.

*attrname* The name of this attribute as a C-string.

*flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).

*val* Pointer to the value.

Definition at line 740 of file ext\_obex\_util.h.

**26.1.4.82** `#define STATIC_ATTR_OBJ_ARRAY(c, attrname, flags, count, vals) STATIC_ATTR_FORMAT(c,attrname,USESVM(object),flags,"O",count,vals)`

Create a shared (static/global) array-of-objects attribute of fixed length, and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- count* The number of items in the [t\\_object\\*](#) array.
- vals* Pointer to the values.

Definition at line 833 of file `ext_obex_util.h`.

**26.1.4.83** `#define STATIC_ATTR_SYM(c, attrname, flags, val) STATIC_ATTR_FORMAT(c,attrname,USESVM(symbol),flags,"s",val)`

Create a shared (static/global) [t\\_symbol\\*](#) attribute and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- val* Pointer to the value.

Definition at line 716 of file `ext_obex_util.h`.

**26.1.4.84** `#define STATIC_ATTR_SYM_ARRAY(c, attrname, flags, count, vals) STATIC_ATTR_FORMAT(c,attrname,USESVM(symbol),flags,"S",count,vals)`

Create a shared (static/global) array-of-symbols attribute of fixed length, and add it to a Max class.

**Parameters:**

- c* The class pointer.
- attrname* The name of this attribute as a C-string.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- count* The number of items in the [t\\_symbol\\*](#) array.
- vals* Pointer to the values.

Definition at line 807 of file `ext_obex_util.h`.

**26.1.4.85** `#define STRUCT_ATTR_ATOM(c, flags, structname, structmember) CLASS_ATTR_ATOM(c,#structmember,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:**

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 425 of file `ext_obex_util.h`.

**26.1.4.86** `#define STRUCT_ATTR_ATOM_ARRAY(c, flags, structname, structmember, size) CLASS_ATTR_ATOM_ARRAY(c,#structmember,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:**

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- size* The number of items in the [t\\_atom](#) array.

Definition at line 525 of file `ext_obex_util.h`.

**26.1.4.87** `#define STRUCT_ATTR_ATOM_VARSIZE(c, flags, structname, structmember, sizemember, maxsize) CLASS_ATTR_ATOM_VARSIZE(c,#structmember,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:**

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- sizemember* The actual number of items in the [t\\_atom](#) array at any given moment.
- maxsize* The maximum number of items in the [t\\_atom](#) array, i.e. the number of members allocated for the array in the struct.

Definition at line 633 of file `ext_obex_util.h`.



#### 26.1.4.88 **#define STRUCT\_ATTR\_CHAR(*c*, *flags*, *structname*, *structmember*) CLASS\_ATTR\_CHAR(*c*,#*structmember*,*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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 360 of file `ext_obex_util.h`.

#### 26.1.4.89 **#define STRUCT\_ATTR\_CHAR\_ARRAY(*c*, *flags*, *structname*, *structmember*, *size*) CLASS\_ATTR\_CHAR\_ARRAY(*c*,#*structmember*,*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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- size* The number of items in the char array.

Definition at line 455 of file `ext_obex_util.h`.

#### 26.1.4.90 **#define STRUCT\_ATTR\_CHAR\_VARSIZE(*c*, *flags*, *structname*, *structmember*, *sizemember*, *maxsize*) CLASS\_ATTR\_CHAR\_VARSIZE(*c*,#*structmember*,*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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- sizemember* The actual number of items in the char array at any given moment.
- maxsize* The maximum number of items in the char array, i.e. the number of members allocated for the array in the struct.

Definition at line 558 of file `ext_obex_util.h`.

#### 26.1.4.91 **#define STRUCT\_ATTR\_DOUBLE(*c*, *flags*, *structname*, *structmember*) CLASS\_ATTR\_DOUBLE(*c*,#*structmember*,*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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 399 of file `ext_obex_util.h`.

#### 26.1.4.92 **#define STRUCT\_ATTR\_DOUBLE\_ARRAY(*c*, *flags*, *structname*, *structmember*, *size*) CLASS\_ATTR\_DOUBLE\_ARRAY(*c*,#*structmember*,*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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- size* The number of items in the double array.

Definition at line 497 of file `ext_obex_util.h`.

#### 26.1.4.93 **#define STRUCT\_ATTR\_DOUBLE\_VARSIZE(*c*, *flags*, *structname*, *structmember*, *sizemember*, *maxsize*) CLASS\_ATTR\_DOUBLE\_VARSIZE(*c*,#*structmember*,*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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- sizemember* The actual number of items in the double array at any given moment.
- maxsize* The maximum number of items in the double array, i.e. the number of members allocated for the array in the struct.

Definition at line 603 of file `ext_obex_util.h`.

#### 26.1.4.94 **#define STRUCT\_ATTR\_FLOAT(c, flags, structname, structmember) CLASS\_ATTR\_FLOAT(c,#structmember,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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 386 of file `ext_obex_util.h`.

#### 26.1.4.95 **#define STRUCT\_ATTR\_FLOAT\_ARRAY(c, flags, structname, structmember, size) CLASS\_ATTR\_FLOAT\_ARRAY(c,#structmember,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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- size* The number of items in the floats array.

Definition at line 483 of file `ext_obex_util.h`.

#### 26.1.4.96 **#define STRUCT\_ATTR\_FLOAT\_VARSIZE(c, flags, structname, structmember, sizemember, maxsize) CLASS\_ATTR\_FLOAT\_VARSIZE(c,#structmember,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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- sizemember* The actual number of items in the float array at any given moment.
- maxsize* The maximum number of items in the float array, i.e. the number of members allocated for the array in the struct.

Definition at line 588 of file `ext_obex_util.h`.

#### 26.1.4.97 **#define STRUCT\_ATTR\_LONG(*c*, *flags*, *structname*, *structmember*) CLASS\_ATTR\_LONG(*c*,#*structmember*,*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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 373 of file `ext_obex_util.h`.

#### 26.1.4.98 **#define STRUCT\_ATTR\_LONG\_ARRAY(*c*, *flags*, *structname*, *structmember*, *size*) CLASS\_ATTR\_LONG\_ARRAY(*c*,#*structmember*,*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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- size* The number of items in the long array.

Definition at line 469 of file `ext_obex_util.h`.

#### 26.1.4.99 **#define STRUCT\_ATTR\_LONG\_VARSIZE(*c*, *flags*, *structname*, *structmember*, *sizemember*, *maxsize*) CLASS\_ATTR\_LONG\_VARSIZE(*c*,#*structmember*,*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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in [e\\_max\\_attrflags](#).
- structname* The C identifier for the struct (containing a valid [t\\_object](#) header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- sizemember* The actual number of items in the long array at any given moment.
- maxsize* The maximum number of items in the long array, i.e. the number of members allocated for the array in the struct.

Definition at line 573 of file `ext_obex_util.h`.

#### 26.1.4.100 `#define STRUCT_ATTR_OBJ(c, flags, structname, structmember) CLASS_ATTR_OBJ(c,#structmember,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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in `e_max_attrflags`.
- structname* The C identifier for the struct (containing a valid `t_object` header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 438 of file `ext_obex_util.h`.

#### 26.1.4.101 `#define STRUCT_ATTR_OBJ_ARRAY(c, flags, structname, structmember, size) CLASS_ATTR_OBJ_ARRAY(c,#structmember,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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in `e_max_attrflags`.
- structname* The C identifier for the struct (containing a valid `t_object` header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- size* The number of items in the `t_object*` array.

Definition at line 539 of file `ext_obex_util.h`.

#### 26.1.4.102 `#define STRUCT_ATTR_OBJ_VARSIZE(c, flags, structname, structmember, sizemember, maxsize) CLASS_ATTR_OBJ_VARSIZE(c,#structmember,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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in `e_max_attrflags`.
- structname* The C identifier for the struct (containing a valid `t_object` header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- sizemember* The actual number of items in the `t_object*` array at any given moment.
- maxsize* The maximum number of items in the `t_object*` array, i.e. the number of members allocated for the array in the struct.

Definition at line 648 of file `ext_obex_util.h`.

#### 26.1.4.103 `#define STRUCT_ATTR_SYM(c, flags, structname, structmember) CLASS_ATTR_SYM(c,#structmember,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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in `e_max_attrflags`.
- structname* The C identifier for the struct (containing a valid `t_object` header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.

Definition at line 412 of file `ext_obex_util.h`.

#### 26.1.4.104 `#define STRUCT_ATTR_SYM_ARRAY(c, flags, structname, structmember, size) CLASS_ATTR_SYM_ARRAY(c,#structmember,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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in `e_max_attrflags`.
- structname* The C identifier for the struct (containing a valid `t_object` header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- size* The number of items in the `t_symbol*` array.

Definition at line 511 of file `ext_obex_util.h`.

#### 26.1.4.105 `#define STRUCT_ATTR_SYM_VARSIZE(c, flags, structname, structmember, sizemember, maxsize) CLASS_ATTR_SYM_VARSIZE(c,#structmember,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:

- c* The class pointer.
- flags* Any flags you wish to declare for this attribute, as defined in `e_max_attrflags`.
- structname* The C identifier for the struct (containing a valid `t_object` header) representing an instance of this class.
- structmember* The C identifier of the member in the struct that holds the value of this attribute.
- sizemember* The actual number of items in the `t_symbol*` array at any given moment.
- maxsize* The maximum number of items in the `t_symbol*` array, i.e. the number of members allocated for the array in the struct.

Definition at line 618 of file `ext_obex_util.h`.

## 26.1.5 Enumeration Type Documentation

### 26.1.5.1 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 *can* 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 *can* be set from C code.

**`ATTR_GET_DEFER`** Any attribute queries will be called through a [defer\(\)](#).

**`ATTR_GET_USURP`** Any calls to query the attribute will be called through the equivalent of a [defer\(\)](#), repeated calls will be ignored until the getter is actually run.

**`ATTR_GET_DEFER_LOW`** Any attribute queries will be called through a [defer\\_low\(\)](#).

**`ATTR_GET_USURP_LOW`** Any calls to query the attribute will be called through the equivalent of a [defer\\_low\(\)](#), repeated calls will be ignored until the getter is actually run.

**`ATTR_SET_DEFER`** The attribute setter will be called through a [defer\(\)](#).

**`ATTR_SET_USURP`** Any calls to set the attribute will be called through the equivalent of a [defer\\_low\(\)](#), repeated calls will be ignored until the setter is actually run.

**`ATTR_SET_DEFER_LOW`** The attribute setter will be called through a [defer\\_low\(\)](#).

**`ATTR_SET_USURP_LOW`** Any calls to set the attribute will be called through the equivalent of a [defer\\_low\(\)](#), repeated calls will be ignored until the setter is actually run.

Definition at line 33 of file `ext_obex.h`.

## 26.1.6 Function Documentation

### 26.1.6.1 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:

***x*** Pointer to the attribute to receive the filter

***min*** Minimum value for the clip filter

**max** Maximum value for the clip filter

**usemin** Sets this value to 0 if the minimum clip value should *not* be used. Otherwise, set the value to non-zero.

**usemax** Sets this value to 0 if the minimum clip value should *not* 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.

**26.1.6.2 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:**

**x** Pointer to the attribute to receive the filter

**scale** Scale value. Data sent to the attribute will be scaled by this amount. Data retrieved from the attribute will be scaled by its reciprocal. *Scaling occurs previous to clipping.*

**min** Minimum value for the clip filter

**max** Maximum value for the clip filter

**usemin** Sets this value to 0 if the minimum clip value should *not* be used. Otherwise, set the value to non-zero.

**usemax** Sets this value to 0 if the minimum clip value should *not* 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.

**26.1.6.3 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:**

**x** Pointer to the attribute to receive the filter

**min** Minimum value for the clip filter

**max** Maximum value for the clip filter

**usemin** Sets this value to 0 if the minimum clip value should *not* be used. Otherwise, set the value to non-zero.

**usemax** Sets this value to 0 if the minimum clip value should *not* 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.

**26.1.6.4 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:**

*x* Pointer to the attribute to receive the filter

*scale* Scale value. Data retrieved from the attribute will be scaled by this amount. *Scaling occurs previous to clipping.*

*min* Minimum value for the clip filter

*max* Maximum value for the clip filter

*usemin* Sets this value to 0 if the minimum clip value should *not* be used. Otherwise, set the value to non-zero.

*usemax* Sets this value to 0 if the minimum clip value should *not* 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.

**26.1.6.5 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:**

*x* Pointer to the attribute to receive the filter

*proc* 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.

#### 26.1.6.6 `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:

*x* Pointer to the attribute to receive the filter

*min* Minimum value for the clip filter

*max* Maximum value for the clip filter

*usemin* Sets this value to 0 if the minimum clip value should *not* be used. Otherwise, set the value to non-zero.

*usemax* Sets this value to 0 if the minimum clip value should *not* 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.

#### 26.1.6.7 `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:

*x* Pointer to the attribute to receive the filter

*scale* Scale value. Data sent to the attribute will be scaled by this amount. *Scaling occurs previous to clipping.*

*min* Minimum value for the clip filter

*max* Maximum value for the clip filter

*usemin* Sets this value to 0 if the minimum clip value should *not* be used. Otherwise, set the value to non-zero.

*usemax* Sets this value to 0 if the minimum clip value should *not* 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.

#### 26.1.6.8 `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:

*x* Pointer to the attribute to receive the filter

*proc* 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;
}
```

#### 26.1.6.9 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:

- x* A dictionary instance pointer.
- ac* The number of atoms to parse in av.
- av* 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);
```

#### 26.1.6.10 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:**

*ac* The count of `t_atoms` in *av*  
*av* 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).

**26.1.6.11 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:**

*x* The object whose attributes will be processed  
*ac* The count of `t_atoms` in *av*  
*av* 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;
}
```

**26.1.6.12 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:**

*x* The object instance pointer.  
*d* The dictionary containing the attributes.

**See also:**

[attr\\_args\\_process\(\)](#)

### 26.1.6.13 `t_object* attr_offset_array_new (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:

**name** A name for the attribute, as a C-string

**type** A `t_symbol *` representing a valid attribute type. At the time of this writing, the valid type-symbols are: `_sym_char` (char), `_sym_long` (long), `_sym_float32` (32-bit float), `_sym_float64` (64-bit float), `_sym_atom` (Max `t_atom` pointer), `_sym_symbol` (Max `t_symbol` pointer), `_sym_pointer` (generic pointer) and `_sym_object` (Max `t_object` pointer).

**size** Maximum number of items that may be in the array.

**flags** 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 `e_max_attrflags`.

**mget** The method to use for the attribute's get functionality. If `mget` is NULL, the default method is used. See the discussion under `attribute_new()`, for more information.

**mset** The method to use for the attribute's set functionality. If `mset` is NULL, the default method is used. See the discussion under `attribute_new()`, for more information.

**offsetcount** Byte offset into the object class's data structure of a long variable describing how many array elements (up to `size`) comprise the data to be referenced by the attribute. Typically, the `calcoffset` macro is used to calculate this offset.

**offset** 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 `calcoffset` 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 d
    ata size * /, 10 / * max * /, 0 / * no flags * /, (method)0L, (method)0L,
    calcoffset(t_myobject, atmcount) / * count * /, calcoffset(t_myobject, atm) / * d
    ata * /);
```

### 26.1.6.14 `t_object* attr_offset_new (char * name, t_symbol * type, long flags, method mget, 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:

**name** A name for the attribute, as a C-string

**type** A [t\\_symbol](#) \* representing a valid attribute type. At the time of this writing, the valid type-symbols are: [\\_sym\\_char](#) (char), [\\_sym\\_long](#) (long), [\\_sym\\_float32](#) (32-bit float), [\\_sym\\_float64](#) (64-bit float), [\\_sym\\_atom](#) (Max [t\\_atom](#) pointer), [\\_sym\\_symbol](#) (Max [t\\_symbol](#) pointer), [\\_sym\\_pointer](#) (generic pointer) and [\\_sym\\_object](#) (Max [t\\_object](#) pointer).

**flags** 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 [e\\_max\\_attrflags](#).

**mget** The method to use for the attribute's get functionality. If mget is NULL, the default method is used. See the discussion under [attribute\\_new\(\)](#), for more information.

**mset** The method to use for the attribute's set functionality. If mset is NULL, the default method is used. See the discussion under [attribute\\_new\(\)](#), for more information.

**offset** 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 [calcoffset](#) 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));
```

#### 26.1.6.15 [t\\_object\\*](#) [attribute\\_new](#) (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:

**name** A name for the attribute, as a C-string

**type** A [t\\_symbol](#) \* representing a valid attribute type. At the time of this writing, the valid type-symbols are: [\\_sym\\_char](#) (char), [\\_sym\\_long](#) (long), [\\_sym\\_float32](#) (32-bit float), [\\_sym\\_float64](#) (64-bit float), [\\_sym\\_atom](#) (Max [t\\_atom](#) pointer), [\\_sym\\_symbol](#) (Max [t\\_symbol](#) pointer), [\\_sym\\_pointer](#) (generic pointer) and [\\_sym\\_object](#) (Max [t\\_object](#) pointer).

**flags** 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 [e\\_max\\_attrflags](#).

**mget** The method to use for the attribute's get functionality. If mget is NULL, the default method is used.

**mset** The method to use for the attribute's set functionality. If mset 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 **a
v);

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 **a
v)
{
    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;
}
```

**26.1.6.16 t\_max\_err object\_addattr (void \*x, t\_object \*attr)**

Attaches an attribute directly to an object.

**Parameters:**

- x* An object to which the attribute should be attached
- attr* The attribute's pointer—this should be a pointer returned from `attribute_new()`, `attr_offset_new()` or `attr_offset_array_new()`.

**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.

**26.1.6.17 void\* object\_attr\_get (void \*x, t\_symbol \*attrname)**

Returns the pointer to an attribute, given its name.

**Parameters:**

*x* Pointer to the object whose attribute is of interest  
*attrname* The attribute's name

**Returns:**

This function returns a pointer to the attribute, if successful, or NULL, if unsuccessful.

**26.1.6.18 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:**

*o* The attribute's parent object  
*name* The attribute's name  
*rect* The address of a valid [t\\_rect](#) 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.

**26.1.6.19 long object\_attr\_getchar\_array (void \* x, t\_symbol \* s, long max, uchar \* 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:**

*x* The attribute's parent object  
*s* The attribute's name  
*max* The number of array elements in *vals*. The function will take care not to overwrite the bounds of the array.  
*vals* 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.



**26.1.6.20** `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:**

*b* The attribute's parent object

*attrname* The attribute's name

*prgba* The address of a valid `t_jrgba` 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.

**26.1.6.21** `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:**

*x* The attribute's parent object

*s* The attribute's name

*max* The number of array elements in *vals*. The function will take care not to overwrite the bounds of the array.

*vals* 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.

**26.1.6.22** `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:**

*x* Pointer to the object whose attribute is of interest

*s* The attribute's name

*argc* Unused

*argv* Unused

#### 26.1.6.23 float object\_attr\_getfloat (void \* *x*, t\_symbol \* *s*)

Retrieves the value of an attribute, given its parent object and name.

**Parameters:**

- x* The attribute's parent object
- s* 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.

#### 26.1.6.24 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:**

- x* The attribute's parent object
- s* The attribute's name
- max* The number of array elements in *vals*. The function will take care not to overwrite the bounds of the array.
- vals* 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.

#### 26.1.6.25 t\_max\_err object\_attr\_getjrjba (void \* *ob*, t\_symbol \* *s*, t\_jrjba \* *c*)

Retrieves the value of a color attribute, given its parent object and name.

**Parameters:**

- ob* The attribute's parent object
- s* The attribute's name
- c* The address of a [t\\_jrjba](#) 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.

**26.1.6.26 long object\_attr\_getlong (void \* *x*, t\_symbol \* *s*)**

Retrieves the value of an attribute, given its parent object and name.

**Parameters:**

- x* The attribute's parent object
- s* 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 db\_query\_silent().

**26.1.6.27 long object\_attr\_getlong\_array (void \* *x*, t\_symbol \* *s*, long *max*, 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:**

- x* The attribute's parent object
- s* The attribute's name
- max* The number of array elements in *vals*. The function will take care not to overwrite the bounds of the array.
- vals* 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.

**26.1.6.28 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:**

- o* The attribute's parent object
- name* The attribute's name
- pt* The address of a valid [t\\_pt](#) 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.

**26.1.6.29** `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:**

- o* The attribute's parent object
- name* The attribute's name
- size* The address of a valid [t\\_size](#) 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.

**26.1.6.30** `t_symbol* object_attr_getsym (void * x, t_symbol * s)`

Retrieves the value of an attribute, given its parent object and name.

**Parameters:**

- x* The attribute's parent object
- s* 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.

**26.1.6.31** `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:**

- x* The attribute's parent object
- s* The attribute's name
- max* The number of array elements in *vals*. The function will take care not to overwrite the bounds of the array.
- vals* Pointer to the first element of a pre-allocated array of [t\\_symbol](#) \*s.

**Returns:**

This function returns the number of elements copied into *vals*.

**26.1.6.32 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:**

*x* Pointer to the object whose attribute is of interest

*methodname* The Max message used to call the attribute's `get` or `set` function. For example, `gensym("mode")` or `gensym("getthresh")`.

*attr* A pointer to a void \*, which will be set to the attribute pointer upon successful completion of the function

*get* 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 `get` function of the attribute.

**Returns:**

This function returns the requested method, if successful, or NULL, if unsuccessful.

**26.1.6.33 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:**

*o* The attribute's parent object

*name* The attribute's name

*rect* The address of a valid `t_rect` 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.

**26.1.6.34 t\_max\_err object\_attr\_setchar\_array (void \* *x*, t\_symbol \* *s*, long *count*, uchar \* *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:**

*x* The attribute's parent object

*s* The attribute's name

*count* The number of array elements in *vals*

*vals* 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.

**26.1.6.35 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:**

*b* The attribute's parent object

*attrname* The attribute's name

*prgba* The address of a valid [t\\_jrgba](#) 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.

**26.1.6.36 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:**

*x* The attribute's parent object

*s* The attribute's name

*count* The number of array elements in *vals*

*vals* 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.

**26.1.6.37 t\_max\_err object\_attr\_setfloat (void \* *x*, t\_symbol \* *s*, 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:**

*x* The attribute's parent object

*s* The attribute's name

*c* 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.

**26.1.6.38 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:**

- x* The attribute's parent object
- s* The attribute's name
- count* The number of array elements in *vals*
- vals* 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.

**26.1.6.39 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:**

- ob* The attribute's parent object
- s* The attribute's name
- c* The address of a [t\\_jrgba](#) 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.

**26.1.6.40 t\_max\_err object\_attr\_setlong (void \* *x*, t\_symbol \* *s*, 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:**

- x* The attribute's parent object
- s* The attribute's name
- c* 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 `db_query_silent()`.

**26.1.6.41 t\_max\_err object\_attr\_setlong\_array (void \* *x*, t\_symbol \* *s*, long *count*, 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:**

- x* The attribute's parent object
- s* The attribute's name
- count* The number of array elements in *vals*
- vals* 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.

**26.1.6.42 t\_max\_err object\_attr\_setparse (t\_object \* *x*, t\_symbol \* *s*, char \* *parsestr*)**

Set an attribute value with one or more atoms parsed from a C-string.

**Parameters:**

- x* The object whose attribute will be set.
- s* The name of the attribute to set.
- parsestr* A C-string to parse into an array of atoms to set the attribute value.

**Returns:**

A Max error code.

**See also:**

[atom\\_setparse\(\)](#)

**26.1.6.43 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:**

- o* The attribute's parent object
- name* The attribute's name
- pt* The address of a valid [t\\_pt](#) 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.



**26.1.6.44 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:**

- o* The attribute's parent object
- name* The attribute's name
- size* The address of a valid [t\\_size](#) 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.

**26.1.6.45 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:**

- x* The attribute's parent object
- s* The attribute's name
- c* A [t\\_symbol](#) \*; 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 `db_view_setquery()`.

**26.1.6.46 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:**

- x* The attribute's parent object
- s* The attribute's name
- count* The number of array elements in *vals*
- vals* Pointer to the first element of an array of [t\\_symbol](#) \*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.

**26.1.6.47 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:**

- x* Pointer to the object whose attribute is of interest
- s* The attribute's name
- argc* The count of arguments in *argv*
- argv* Array of t\_atoms; 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.

**26.1.6.48 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:**

- x* Pointer to the object whose attribute is of interest
- s* 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.

**26.1.6.49 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:**

- x* Pointer to the object whose attribute is of interest
- s* The attribute's name

**Returns:**

This function returns 1 if the attribute can be set from the Max interface. Otherwise, it returns 0.

**26.1.6.50 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:**

*x* The object to which the attribute is attached  
*attrsym* 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.

**26.1.6.51 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:**

*x* The object to which the attribute is attached  
*attrsym* 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.

**26.1.6.52 void\* object\_new\_parse (t\_symbol \* *name\_space*, t\_symbol \* *classname*, 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:**

*name\_space* The namespace in which to create the instance. Typically this is either [CLASS\\_BOX](#) or [CLASS\\_NOBOX](#).  
*classname* The name of the class to instantiate.  
*parsestr* A C-string to parse into an array of atoms to set the attribute value.

**Returns:**

A pointer to the new instance.

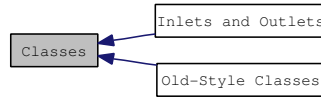
**See also:**

[atom\\_setparse\(\)](#)  
[object\\_new\\_typed\(\)](#)

## 26.2 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 main() function.

Collaboration diagram for Classes:



### Data Structures

- struct [t\\_class](#)

*The data structure for a Max class.*

### Modules

- [Old-Style Classes](#)
- [Inlets and Outlets](#)

*Routines for creating and communicating with inlets and outlets.*

### Defines

- #define [CLASS\\_BOX](#) gensym("box")

*The namespace for all Max object classes which can be instantiated in a box, i.e.*

- #define [CLASS\\_NOBOX](#) gensym("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](#) = 0x00000001L,  
[CLASS\\_FLAG\\_POLYGLOT](#) = 0x00000002L,  
[CLASS\\_FLAG\\_NEWDICTIONARY](#) = 0x00000004L,  
[CLASS\\_FLAG\\_REGISTERED](#) = 0x00000008L,  
[CLASS\\_FLAG\\_UIOBJECT](#) = 0x00000010L,  
[CLASS\\_FLAG\\_ALIAS](#) = 0x00000020L,  
[CLASS\\_FLAG\\_SCHED\\_PURGE](#) = 0x00000040L,  
[CLASS\\_FLAG\\_DO\\_NOT\\_PARSE\\_ATTR\\_ARGS](#) = 0x00000080L,  
[CLASS\\_FLAG\\_NOATTRIBUTES](#) = 0x00010000L,  
[CLASS\\_FLAG\\_OWNATTRIBUTES](#) = 0x00020000L }

*Class flags.*

## Functions

- `t_class * class_new` (char \*name, `method` mnew, `method` mfree, long size, `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, `method` m, 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_casefree` (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.*

### 26.2.1 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 `main()` function. Thus, Max classes are typically defined in the `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.

### 26.2.2 Define Documentation

#### 26.2.2.1 `#define CLASS_BOX gensym("box")`

The namespace for all Max object classes which can be instantiated in a box, i.e. in a patcher.

Definition at line 19 of file `ext_obex.h`.

### 26.2.3 Enumeration Type Documentation

#### 26.2.3.1 `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\_SCHED\_PURGE*** for classes that have called `clock_new()` or `qelem_new()` (don't need to set this yourself)

***CLASS\_FLAG\_DO\_NOT\_PARSE\_ATTR\_ARGS*** override dictionary based constructor attr arg parsing

***CLASS\_FLAG\_NOATTRIBUTES*** for efficiency

***CLASS\_FLAG\_OWATTRIBUTES*** for classes which support a custom attr interface (e.g. jitter)

Definition at line 176 of file `ext_mess.h`.

## 26.2.4 Function Documentation

### 26.2.4.1 `t_max_err class_addattr (t_class * c, t_object * attr)`

Adds an attribute to a previously defined object class.

**Parameters:**

*c* The class pointer

*attr* The attribute to add. The attribute will be a pointer returned by [attribute\\_new\(\)](#), [attr\\_offset\\_new\(\)](#) or [attr\\_offset\\_array\\_new\(\)](#).

**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.

### 26.2.4.2 `t_max_err class_addmethod (t_class * c, method m, char * name, ...)`

Adds a method to a previously defined object class.

**Parameters:**

*c* The class pointer

*m* Function to be called when the method is invoked

*name* 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.

### 26.2.4.3 `t_max_err class_alias (t_class * c, t_symbol * aliasname)`

Registers an alias for a previously defined object class.

**Parameters:**

*c* The class pointer

*aliasname* 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.

#### 26.2.4.4 `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:**

*c* 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.

#### 26.2.4.5 `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:**

*name\_space* The desired class's name space. Typically, either the constant `CLASS_BOX`, for obex classes which can instantiate inside of a Max patcher (e.g. boxes, UI objects, etc.), or the constant `CLASS_NOBOX`, for classes which will only be used internally. Developers can define their own name spaces as well, but this functionality is currently undocumented.

*classname* The name of the class to be looked up

**Returns:**

If successful, this function returns the class's data pointer. Otherwise, it returns NULL.

#### 26.2.4.6 `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:**

*name\_space* The desired class's name space. Typically, either the constant `CLASS_BOX`, for obex classes which can instantiate inside of a Max patcher (e.g. boxes, UI objects, etc.), or the constant `CLASS_NOBOX`, for classes which will only be used internally. Developers can define their own name spaces as well, but this functionality is currently undocumented.

*classname* 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.

#### 26.2.4.7 `t_max_err class_free (t_class * c)`

Frees a previously defined object class. *This function is not typically used by external developers.*

**Parameters:**

*c* 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.

**26.2.4.8 long class\_is\_ui (t\_class \* c)**

Determine if a class is a user interface object.

**Parameters:**

*c* The class pointer.

**Returns:**

True is the class defines a user interface object, otherwise false.

**26.2.4.9 t\_symbol\* class\_nameget (t\_class \* c)**

Retrieves the name of a class, given the class's pointer.

**Parameters:**

*c* The class pointer

**Returns:**

If successful, this function returns the name of the class as a [t\\_symbol](#) \*.

**26.2.4.10 t\_class\* class\_new (char \* name, method mnew, method mfree, long size, 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:**

*name* The class's name, as a C-string

*mnew* The instance creation function

*mfree* The instance free function

*size* The size of the object's data structure in bytes. Usually you use the C sizeof operator here.

*mmenu* The function called when the user creates a new object of the class from the Patch window's palette (UI objects only). Pass 0L if you're not defining a UI object.

*type* A standard Max *type list* 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. *Generally, obex objects have a single type argument, [A\\_GIMME](#), followed by a 0.*

**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.*

#### 26.2.4.11 `long class_obexoffset_get (t_class * c)`

Retrieves the byte-offset of the obex member of the class's data structure.

**Parameters:**

*c* The class pointer

**Returns:**

This function returns the byte-offset of the obex member of the class's data structure.

#### 26.2.4.12 `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:**

*c* The class pointer

*offset* The byte-offset to the obex member of the object's data structure. Conventionally, the macro `calcoffset` is used to calculate the offset.

#### 26.2.4.13 `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:**

*name\_space* The desired class's name space. Typically, either the constant `CLASS_BOX`, for obex classes which can instantiate inside of a Max patcher (e.g. boxes, UI objects, etc.), or the constant `CLASS_NOBOX`, for classes which will only be used internally. Developers can define their own name spaces as well, but this functionality is currently undocumented.

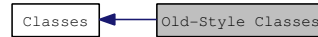
*c* The class pointer

**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.

## 26.3 Old-Style Classes

Collaboration diagram for Old-Style Classes:



### Functions

- void **setup** (**t\_messlist** \*\*ident, **method** makefun, **method** freefun, short 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 **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.*
- void \* **newobject** (void \*maxclass)  
*Use **newobject** to allocate the space for an instance of your class and initialize its object header.*
- void **freeobject** (**t\_object** \*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 you object's name with it's filename on disk.*
- void \* **typedmess** (**t\_object** \*op, **t\_symbol** \*msg, short argc, **t\_atom** \*argp)

*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.*

## 26.3.1 Function Documentation

### 26.3.1.1 void addbang (method *f*)

Used to bind a function to the common triggering message bang.

**Parameters:**

*f* Function to be the bang method.

### 26.3.1.2 void addfloat (method *f*)

Use `addfloat()` to bind a function to the float message received in the leftmost inlet.

**Parameters:**

*f* Function to be the int method.

### 26.3.1.3 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:**

*f* Function to be the float method.

*n* 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](#).

### 26.3.1.4 void addint (method *f*)

Use `addint()` to bind a function to the int message received in the leftmost inlet.

**Parameters:**

*f* Function to be the int method.

### 26.3.1.5 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:**

- f* Function to be the int method.
- n* 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](#).

### 26.3.1.6 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.

**Parameters:**

- f* Function you want to be the method.
- s* C string defining the message.
- type* The first of one or more integers from [e\\_max\\_atomtypes](#) specifying the arguments to the message.
- ... Any additional types from [e\\_max\\_atomtypes](#) for additional arguments.

**See also:**

[Anatomy of a Max Object](#)

### 26.3.1.7 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:**

- name* An alternative name for the user to use to make an object of your class.

### 26.3.1.8 void class\_setname (char \* *obname*, char \* *filename*)

Use [class\\_setname\(\)](#) to associate you object's name with it's filename on disk.

**Parameters:**

- obname* A character string with the name of your object class as it appears in Max.
- filename* A character string with the name of your external's file as it appears on disk.

### 26.3.1.9 method egetfn (t\_object \* op, t\_symbol \* msg)

Use [egetfn\(\)](#) to send an untyped message to a Max object that always works.

#### Parameters:

*op* Receiver of the message.

*msg* 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.

### 26.3.1.10 void freeobject (t\_object \* 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:

*op* The object instance pointer to free.

#### Remarks:

This function can be replaced by the use of [object\\_free\(\)](#). Unlike [freeobject\(\)](#), [object\\_free\(\)](#) checks to make sure the pointer is not NULL before trying to free it.

#### See also:

[newobject\(\)](#)  
[object\\_free\(\)](#)

### 26.3.1.11 method getfn (t\_object \* op, t\_symbol \* msg)

Use [getfn\(\)](#) to send an untyped message to a Max object with error checking.

#### Parameters:

*op* Receiver of the message.

*msg* 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.

**26.3.1.12 void\* newinstance (t\_symbol \* s, short argc, t\_atom \* argv)**

Make a new instance of an existing Max class.

**Parameters:**

*s* className Symbol specifying the name of the class of the instance to be created.

*argc* Count of arguments in argv.

*argv* Array of t\_atoms; 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.

**26.3.1.13 void\* newobject (void \* maxclass)**

Use newobject to allocate the space for an instance of your class and initialize its object header.

**Parameters:**

*maxclass* 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.

**26.3.1.14 void setup (t\_messlist \*\* ident, method makefun, method freefun, short 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:**

*ident* A global variable in your code that points to the initialized class.

**makefun** Your instance creation function.

**freefun** Your instance free function (see Chapter 7).

**size** The size of your objects data structure in bytes. Usually you use the C sizeof operator here.

**menufun** No longer used. You should pass NULL for this parameter.

**type** 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 [e\\_max\\_atomtypes](#). The final argument of the type list should be a 0.

See also:

[Anatomy of a Max Object](#)

### 26.3.1.15 void\* typedmess (t\_object \* op, t\_symbol \* msg, short argc, t\_atom \* argp)

Send a typed message directly to a Max object.

**Parameters:**

**op** Max object that will receive the message.

**msg** The message selector.

**argc** Count of message arguments in argv.

**argp** 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);
```

### 26.3.1.16 method zgetfn (t\_object \* op, t\_symbol \* msg)

Use [zgetfn\(\)](#) to send an untyped message to a Max object without error checking.

**Parameters:**

**op** Receiver of the message.

**msg** Message selector.

**Returns:**

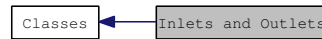
zgetfn returns a pointer to the method bound to the message selector msg in the receiver's message list. It returns 0 but doesn't print an error message in Max Window if the method can't be found.



## 26.4 Inlets and Outlets

Routines for creating and communicating with inlets and outlets.

Collaboration diagram for Inlets and Outlets:



### Functions

- void \* [inlet\\_new](#) (void \*x, 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, 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](#) (void \*o)  
*Use [outlet\\_bang\(\)](#) to send a bang message out an outlet.*
- void \* [outlet\\_int](#) (void \*o, long n)  
*Use [outlet\\_int\(\)](#) to send a float message out an outlet.*
- void \* [outlet\\_float](#) (void \*o, double f)  
*Use [outlet\\_float\(\)](#) to send an int message out an outlet.*
- void \* [outlet\\_list](#) (void \*o, [t\\_symbol](#) \*s, short ac, [t\\_atom](#) \*av)  
*Use [outlet\\_list\(\)](#) to send a list message out an outlet.*
- void \* [outlet\\_anything](#) (void \*o, [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.*

## 26.4.1 Detailed Description

Routines for creating and communicating with inlets and outlets.

## 26.4.2 Function Documentation

### 26.4.2.1 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.

### 26.4.2.2 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.

### 26.4.2.3 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.

**26.4.2.4 void\* inlet\_new (void \* x, 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\(\)](#) ceates 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](#).

**26.4.2.5 void\* intin (void \* x, short n)**

Use [intin\(\)](#) to create an inlet typed to receive only integers.

**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.

**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.

#### 26.4.2.6 void\* intout (void \* *x*)

Use [intout\(\)](#) to create an outlet that will always send the int 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.

#### 26.4.2.7 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.

#### 26.4.2.8 void\* outlet\_anything (void \* *o*, t\_symbol \* *s*, short *ac*, t\_atom \* *av*)

Use [outlet\\_anything\(\)](#) to send any message out an outlet.

**Parameters:**

- o* Outlet that will send the message.
- s* The message selector [t\\_symbol\\*](#).
- ac* Number of elements in the list in argv.
- av* 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.

**26.4.2.9 void\* outlet\_bang (void \* *o*)**

Use `outlet_bang()` to send a bang message out an outlet.

**Parameters:**

*o* Outlet that will send the message.

**Returns:**

Returns 0 if a stack overflow occurred, otherwise returns 1.

**26.4.2.10 void\* outlet\_float (void \* *o*, double *f*)**

Use `outlet_float()` to send an int message out an outlet.

**Parameters:**

*o* Outlet that will send the message.

*f* Float value to send.

**Returns:**

Returns 0 if a stack overflow occurred, otherwise returns 1.

**26.4.2.11 void\* outlet\_int (void \* *o*, long *n*)**

Use `outlet_int()` to send a float message out an outlet.

**Parameters:**

*o* Outlet that will send the message.

*n* Integer value to send.

**Returns:**

Returns 0 if a stack overflow occurred, otherwise returns 1.

#### 26.4.2.12 void\* outlet\_list (void \*o, t\_symbol \*s, short ac, t\_atom \*av)

Use [outlet\\_list\(\)](#) to send a list message out an outlet.

##### Parameters:

- o* Outlet that will send the message.
- s* Should be NULL, but can be the `_sym_list`.
- ac* Number of elements in the list in `argv`.
- av* 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.

#### 26.4.2.13 void\* outlet\_new (void \*x, char \*s)

Use [outlet\\_new\(\)](#) to create an outlet that can send a specific non-standard message, or any message.

##### Parameters:

- x* Your object.
- s* 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.

#### 26.4.2.14 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:

*master* Your object.

##### Returns:

The index number of the inlet that received the message.

#### 26.4.2.15 void\* proxy\_new (void \* *x*, long *id*, long \* *stuffloc*)

Use proxy\_new to create a new Proxy object.

##### Parameters:

*x* Your object.

*id* A non-zero number to be written into your object when a message is received in this particular Proxy. Normally, id will be the inlet “number” analogous to in1, in2 etc.

*stuffloc* A pointer to a location where the id value will be written.

##### Returns:

A pointer to the new proxy inlet.

##### Remarks:

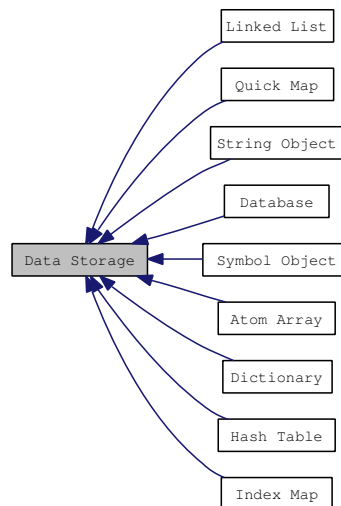
This routine creates a new Proxy object (that includes an inlet). It allows you to identify messages based on an id value stored in the location specified by stuffLoc. You should store the pointer returned by [proxy\\_new\(\)](#) because you’ll need to free all Proxies in your object’s free function 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.

## 26.5 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 support currently consists of a SQLite ( <http://sqlite.org> ) extension which is loaded dynamically by Max at launch time.*

- [Dictionary](#)

*In Max 5, we have a new "dictionary" object which can be used for object prototypes, object serialization, object constructors, and other tasks.*

- [Hash Table](#)

*Max's hashtable object implements a hash table ( [http://en.wikipedia.org/wiki/Hash\\_table](http://en.wikipedia.org/wiki/Hash_table) ).*

- [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](#)

*Max's linklist object implements a doubly-linked-list ( [http://en.wikipedia.org/wiki/Linked\\_list](http://en.wikipedia.org/wiki/Linked_list) ) together with a high-level interface for manipulating and accessing values in the list.*

- [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.*

## Typedefs

- `typedef long(* t\_cmpfn )(void *, void *)`

*Comparison function pointer type.*

## Enumerations

- `enum e\_max\_datastore\_flags {`  
`OBJ\_FLAG\_OBJ = 0x00000000,`  
`OBJ\_FLAG\_REF = 0x00000001,`  
`OBJ\_FLAG\_DATA = 0x00000002,`  
`OBJ\_FLAG\_MEMORY = 0x00000004,`  
`OBJ\_FLAG\_SILENT = 0x00000100 }`

*Flags used in linklist and hashtab objects.*

### 26.5.1 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.

### 26.5.2 Typedef Documentation

#### 26.5.2.1 `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\(\)](#)

Definition at line 63 of file `ext_linklist.h`.

## 26.5.3 Enumeration Type Documentation

### 26.5.3.1 enum e\_max\_datastore\_flags

Flags used in linklist and hashtable objects.

**Enumerator:**

*OBJ\_FLAG\_OBJ* free using [object\\_free\(\)](#)

*OBJ\_FLAG\_REF* don't free

*OBJ\_FLAG\_DATA* don't free data or call method

*OBJ\_FLAG\_MEMORY* don't call method, and when freeing use [systemem\\_freeptr\(\)](#) instead of freeobject

*OBJ\_FLAG\_SILENT* don't notify when modified

Definition at line 63 of file ext\_obex.h.

## 26.6 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.*

### Functions

- [t\\_atomarray \\* atomarray\\_new](#) (long ac, [t\\_atom](#) \*av)  
*Create a new atomarray object.*
- [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.*
- 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\\_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.*

## 26.6.1 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](#)

## 26.6.2 Function Documentation

### 26.6.2.1 void atomarray\_appendatom ([t\\_atomarray](#) \*x, [t\\_atom](#) \*a)

Copy a new atom onto the end of the array.

**Parameters:**

*x* The atomarray instance.

*a* A pointer to the new atom to append to the end of the array.

See also:

[atomarray\\_appendatoms\(\)](#)  
[atomarray\\_setatoms\(\)](#)

### 26.6.2.2 void atomarray\_appendatoms ([t\\_atomarray](#) \*x, long ac, [t\\_atom](#) \*av)

Copy multiple new atoms onto the end of the array.

**Parameters:**

*x* The atomarray instance.

*ac* The number of new atoms to be appended to the array.

*av* A pointer to the first of the new atoms to append to the end of the array.

See also:

[atomarray\\_appendatom\(\)](#)  
[atomarray\\_setatoms\(\)](#)

**26.6.2.3 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:**

*x* The atomarray instance.

*index* The zero-based index of the atom to remove from the array.

**26.6.2.4 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:**

*x* The atomarray instance.

**Returns:**

The number of atoms in the array.

**26.6.2.5 t\_max\_err atomarray\_copyatoms (t\_atomarray \* *x*, long \* *ac*, t\_atom \*\* *av*)**

Retrieve a copy of the atoms in the array. 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 [t\\_atom](#) 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\(\)](#)

### 26.6.2.6 void\* atomarray\_duplicate (t\_atomarray \* x)

Create a new atomarray object which is a copy of another atomarray object.

#### Parameters:

*x* The atomarray instance which is to be copied.

#### Returns:

A new atomarray which is copied from *x*.

#### See also:

[atomarray\\_new\(\)](#)

### 26.6.2.7 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\(\)](#)

### 26.6.2.8 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 `t_atom` pointer where the address of the first atom of the array will be set.

**Returns:**

A Max error code.

**See also:**

`atomarray_copyatoms()`

**26.6.2.9 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...
}
```

**26.6.2.10 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.

**26.6.2.11 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 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\(\)](#)

**26.6.2.12 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:**

- x* The atomarray instance.
- 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:**

A Max error code.



## 26.7 Database

Max's database support currently consists of a SQLite ( <http://sqlite.org> ) extension which is loaded dynamically by Max at launch time.

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_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_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.*
- `unsigned long 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\\_stringtodate](#) (const char \*string, unsigned long \*date)

*A utility to convert from a sql datetime string into seconds.*

- void [db\\_util\\_datetosting](#) (const unsigned long date, char \*string)

*A utility to convert from seconds into a sql-ready datetime string.*

### 26.7.1 Detailed Description

Max's 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.

### 26.7.2 Typedef Documentation

#### 26.7.2.1 typedef t\_object t\_database

A database object. Use [db\\_open\(\)](#) and [db\\_close\(\)](#) to create and free database objects.

Definition at line 23 of file [ext\\_database.h](#).

#### 26.7.2.2 typedef t\_object t\_db\_result

A database result object. This is what the database object returns when a query is executed.

Definition at line 29 of file [ext\\_database.h](#).

#### 26.7.2.3 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.

Definition at line 35 of file [ext\\_database.h](#).

### 26.7.3 Function Documentation

#### 26.7.3.1 t\_max\_err db\_close (t\_database \*\* db)

Close an open database.

##### Parameters:

- db* The address of the [t\\_database](#) pointer for your database instance. The pointer will be freed and set NULL upon return.

**Returns:**

An error code.

Definition at line 70 of file ext\_database.c.

References MAX\_ERR\_NONE, and object\_free().

Here is the call graph for this function:



### 26.7.3.2 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:**

***dbname*** The name of the database.

***fullpath*** 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.

***db*** The address of a [t\\_database](#) 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.

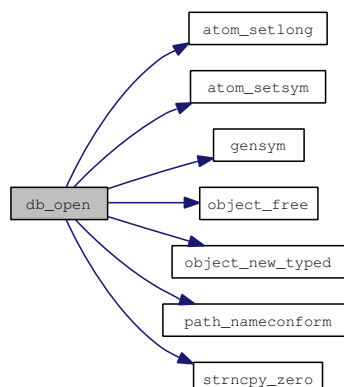
**Returns:**

An error code.

Definition at line 18 of file ext\_database.c.

References atom\_setlong(), atom\_setsym(), gensym(), MAX\_ERR\_GENERIC, MAX\_ERR\_NONE, MAX\_PATH\_CHARS, object\_free(), object\_new\_typed(), path\_nameconform(), PATH\_TYPE\_ABSOLUTE, and strncpy\_zero().

Here is the call graph for this function:



### 26.7.3.3 `t_max_err db_query (t_database * db, t_db_result ** dbresult, const char * sql, ...)`

Execute a SQL query on the database.

#### Parameters:

**db** The `t_database` pointer for your database instance.

**dbresult** The address of a `t_db_result` 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 `object_free()`.

**sql** 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:

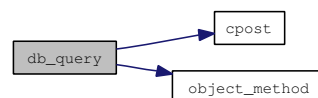
An error code.

Definition at line 78 of file `ext_database.c`.

References `cpost()`, `MAX_ERR_GENERIC`, and `object_method()`.

Referenced by `db_query_silent()`, and `db_query_table_new()`.

Here is the call graph for this function:



### 26.7.3.4 `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:

**db** The `t_database` pointer for your database instance.

**id** The address of a variable to hold the result on return.

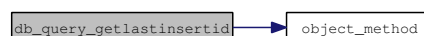
#### Returns:

An error code.

Definition at line 122 of file `ext_database.c`.

References `MAX_ERR_NONE`, and `object_method()`.

Here is the call graph for this function:



### 26.7.3.5 `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:

**db** The `t_database` pointer for your database instance.

**dbresult** The address of a `t_db_result` 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 `object_free()`.

**sql** 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:

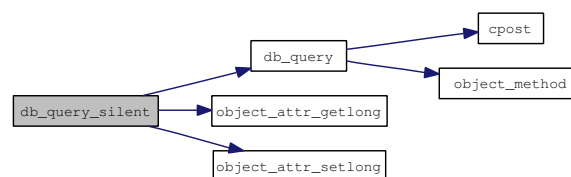
An error code.

Definition at line 104 of file `ext_database.c`.

References `db_query()`, `object_attr_getlong()`, and `object_attr_setlong()`.

Referenced by `db_query_table_addcolumn()`.

Here is the call graph for this function:



### 26.7.3.6 `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:

**db** The `t_database` pointer for your database instance.

**tablename** The name of the table to which the column should be added.

**columnname** The name to use for the new column.

**columntype** The SQL type for the data that will be stored in the column. For example: "INTEGER" or "VARCHAR"

**flags** If you wish to specify any additional information for the column, then pass that here. Otherwise pass NULL.

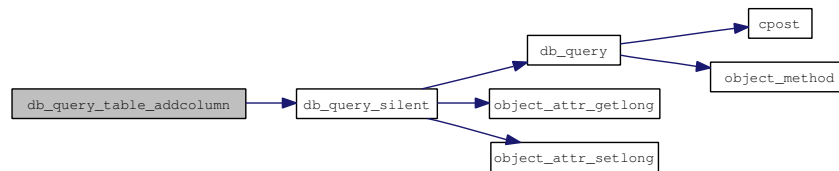
#### Returns:

An error code.

Definition at line 143 of file ext\_database.c.

References db\_query\_silent().

Here is the call graph for this function:



### 26.7.3.7 t\_max\_err db\_query\_table\_new (t\_database \* db, const char \* tablename)

Create a new table in a database.

#### Parameters:

**db** The [t\\_database](#) pointer for your database instance.

**tablename** 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.

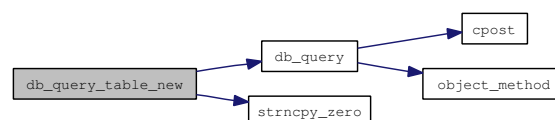
#### Returns:

An error code.

Definition at line 129 of file ext\_database.c.

References db\_query(), and strncpy\_zero().

Here is the call graph for this function:



### 26.7.3.8 void db\_result\_clear (t\_db\_result \* result)

Zero-out a database result.

#### Parameters:

**result** The [t\\_db\\_result](#) pointer for your query results.

Definition at line 226 of file ext\_database.c.

References object\_method().

Here is the call graph for this function:



### 26.7.3.9 unsigned long 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. The value will be coerced from an expected datetime field into seconds.

#### Parameters:

**result** The [t\\_db\\_result](#) pointer for your query results.

**recordindex** The zero-based index number of the record (row) in the result.

**fieldindex** The zero-based index number of the field (column) in the result.

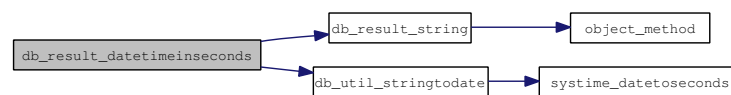
#### Returns:

The datetime represented in seconds.

Definition at line 277 of file ext\_database.c.

References [db\\_result\\_string\(\)](#), and [db\\_util\\_stringtodate\(\)](#).

Here is the call graph for this function:



### 26.7.3.10 char\* db\_result\_fieldname (t\_db\_result \* result, long fieldindex)

Return the name of a field specified by its index number.

#### Parameters:

**result** The [t\\_db\\_result](#) pointer for your query results.

**fieldindex** The zero-based index number of the field (column) in the result.

#### Returns:

A C-String with the name of the field.

Definition at line 247 of file ext\_database.c.

References [object\\_method\(\)](#).

Here is the call graph for this function:



### 26.7.3.11 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:**

*result* The [t\\_db\\_result](#) pointer for your query results.

*recordindex* The zero-based index number of the record (row) in the result.

*fieldindex* 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.

Definition at line 267 of file ext\_database.c.

References [object\\_method\(\)](#).

Here is the call graph for this function:

**26.7.3.12 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:**

*result* The [t\\_db\\_result](#) pointer for your query results.

*recordindex* The zero-based index number of the record (row) in the result.

*fieldindex* 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.

Definition at line 257 of file ext\_database.c.

References [object\\_method\(\)](#).

Here is the call graph for this function:

**26.7.3.13 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:**

*result* The [t\\_db\\_result](#) 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\(\)](#).

Definition at line 213 of file ext\_database.c.

References [object\\_method\(\)](#).

Here is the call graph for this function:

**26.7.3.14 long db\_result\_numfields (t\_db\_result \* *result*)**

Return a count of all fields (columns) in the query result.

**Parameters:**

*result* The [t\\_db\\_result](#) pointer for your query results.

**Returns:**

The count of fields in the query result.

Definition at line 239 of file ext\_database.c.

References [object\\_method\(\)](#).

Here is the call graph for this function:

**26.7.3.15 long db\_result\_numrecords (t\_db\_result \* *result*)**

Return a count of all records in the query result.

**Parameters:**

*result* The [t\\_db\\_result](#) pointer for your query results.

**Returns:**

The count of records in the query result.

Definition at line 231 of file ext\_database.c.

References [object\\_method\(\)](#).

Here is the call graph for this function:



**26.7.3.16 void db\_result\_reset (t\_db\_result \* *result*)**

Reset the interface for walking a result's record list to the first record.

**Parameters:**

*result* The [t\\_db\\_result](#) pointer for your query results.

Definition at line 221 of file ext\_database.c.

References [object\\_method\(\)](#).

Here is the call graph for this function:

**26.7.3.17 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:**

*result* The [t\\_db\\_result](#) pointer for your query results.

*recordindex* The zero-based index number of the record (row) in the result.

*fieldindex* 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.

Definition at line 252 of file ext\_database.c.

References [object\\_method\(\)](#).

Referenced by [db\\_result\\_datetimeinseconds\(\)](#).

Here is the call graph for this function:

**26.7.3.18 t\_max\_err db\_transaction\_end (t\_database \* *db*)**

Finalize a database transaction.

**Parameters:**

*db* The [t\\_database](#) pointer for your database instance.

**Returns:**

An error code.

Definition at line 157 of file ext\_database.c.

References object\_method().

Here is the call graph for this function:



### 26.7.3.19 t\_max\_err db\_transaction\_flush (t\_database \* db)

Force any open transactions to close.

#### Parameters:

*db* The [t\\_database](#) pointer for your database instance.

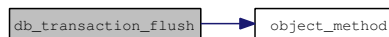
#### Returns:

An error code.

Definition at line 162 of file ext\_database.c.

References object\_method().

Here is the call graph for this function:



### 26.7.3.20 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:

*db* The [t\\_database](#) pointer for your database instance.

#### Returns:

An error code.

Definition at line 152 of file ext\_database.c.

References object\_method().

Here is the call graph for this function:



**26.7.3.21 void db\_util\_datetosting (const unsigned long *date*, char \* *string*)**

A utility to convert from seconds into a sql-ready datetime string.

**Parameters:**

*date* The datetime represented in seconds.

*string* The address of a valid C-string whose contents will be set to a SQL-ready string format upon return.

Definition at line 300 of file ext\_database.c.

References `t_datetime::day`, `t_datetime::hour`, `t_datetime::minute`, `t_datetime::month`, `t_datetime::second`, `systeme_secondstodate()`, and `t_datetime::year`.

Here is the call graph for this function:

**26.7.3.22 void db\_util\_stringtodate (const char \* *string*, unsigned long \* *date*)**

A utility to convert from a sql datetime string into seconds.

**Parameters:**

*string* A C-string containing a date and time in SQL format.

*date* The datetime represented in seconds upon return.

Definition at line 290 of file ext\_database.c.

References `t_datetime::day`, `t_datetime::hour`, `t_datetime::minute`, `t_datetime::month`, `t_datetime::second`, `systeme_datetoseconds()`, and `t_datetime::year`.

Referenced by `db_result_datetimeinseconds()`.

Here is the call graph for this function:

**26.7.3.23 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:**

*db* The [t\\_database](#) pointer for your database instance.

*sql* A SQL query that defines the set of results provided by the view.

***dbview*** The address of a NULL [t\\_db\\_view](#) pointer which will be set with the new view upon return.

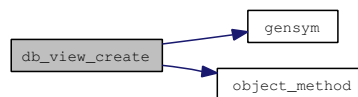
**Returns:**

An error code.

Definition at line 170 of file ext\_database.c.

References [gensym\(\)](#), [MAX\\_ERR\\_GENERIC](#), [MAX\\_ERR\\_NONE](#), and [object\\_method\(\)](#).

Here is the call graph for this function:



### 26.7.3.24 `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:**

***dbview*** The [t\\_db\\_view](#) pointer for your database view instance.

***result*** The address of a pointer to a [t\\_db\\_result](#) object. This pointer will be overwritten with the view's result pointer upon return.

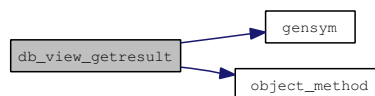
**Returns:**

An error code.

Definition at line 193 of file ext\_database.c.

References [gensym\(\)](#), [MAX\\_ERR\\_GENERIC](#), [MAX\\_ERR\\_NONE](#), and [object\\_method\(\)](#).

Here is the call graph for this function:



### 26.7.3.25 `t_max_err db_view_remove (t_database * db, t_db_view ** dbview)`

Remove a database view created using [db\\_view\\_create\(\)](#).

**Parameters:**

***db*** The [t\\_database](#) pointer for your database instance for which this view was created.

***dbview*** The address of the [t\\_db\\_view](#) pointer for the view. This pointer will be freed and set NULL upon return.

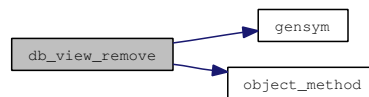
**Returns:**

An error code.

Definition at line 185 of file ext\_database.c.

References gensym(), MAX\_ERR\_NONE, and object\_method().

Here is the call graph for this function:

**26.7.3.26 t\_max\_err db\_view\_setquery (t\_db\_view \* dbview, char \* newquery)**

Set the query used by the view.

**Parameters:**

**dbview** The `t_db_view` pointer for your database view instance.

**newquery** The SQL string to define a new query for the view, replacing the old query.

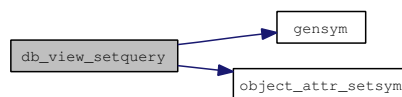
**Returns:**

An error code.

Definition at line 202 of file ext\_database.c.

References gensym(), MAX\_ERR\_GENERIC, and object\_attr\_setsym().

Here is the call graph for this function:



## 26.8 Dictionary

In Max 5, we have a new "dictionary" object which can be used for object prototypes, object serialization, object constructors, and other tasks.

Collaboration diagram for Dictionary:



### Data Structures

- struct `t_dictionary_entry`  
*A dictionary entry.*
- struct `t_dictionary`  
*The dictionary object.*

### Functions

- `t_dictionary * dictionary_new ()`  
*Create a new linklist object.*
- `t_max_err dictionary_appendlong (t_dictionary *d, t_symbol *key, 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 (t_dictionary *d, t_symbol *key, long *value)`  
*Retrieve a long integer from the dictionary.*
- `t_max_err dictionary_getfloat (t_dictionary *d, t_symbol *key, double *value)`  
*Retrieve a double-precision float from the dictionary.*
- `t_max_err dictionary_getsym (t_dictionary *d, t_symbol *key, t_symbol **value)`  
*Retrieve a `t_symbol*` from the dictionary.*
- `t_max_err dictionary_getatom (t_dictionary *d, t_symbol *key, t_atom *value)`  
*Copy a `t_atom` from the dictionary.*
- `t_max_err dictionary_getstring (t_dictionary *d, t_symbol *key, const char **value)`  
*Retrieve a C-string pointer from the dictionary.*
- `t_max_err dictionary_getatoms (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_copyatoms (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 (t_dictionary *d, t_symbol *key, t_object **value)`  
*Retrieve a `t_atomarray` pointer from the dictionary.*
- `t_max_err dictionary_getdictionary (t_dictionary *d, t_symbol *key, t_object **value)`  
*Retrieve a `t_dictionary` pointer from the dictionary.*
- `t_max_err dictionary_getobject (t_dictionary *d, t_symbol *key, t_object **value)`  
*Retrieve a `t_object` pointer from the dictionary.*
- `long dictionary_entryisstring (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 (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 (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 (t_dictionary *d, t_symbol *key)`  
*Test a key to set if it exists in the dictionary.*
- `long dictionary_getentrycount (t_dictionary *d)`  
*Return the number of keys in a dictionary.*
- `t_max_err dictionary_getkeys (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.*
- `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` (`t_dictionary` \*d, `t_symbol` \*key, long \*value, long def)  
*Retrieve a long integer from the dictionary.*
- `t_max_err` `dictionary_getdeffloat` (`t_dictionary` \*d, `t_symbol` \*key, double \*value, double def)  
*Retrieve a double-precision float from the dictionary.*
- `t_max_err` `dictionary_getdefsymb` (`t_dictionary` \*d, `t_symbol` \*key, `t_symbol` \*\*value, `t_symbol` \*def)  
*Retrieve a `t_symbol`\* from the dictionary.*
- `t_max_err` `dictionary_getdefatom` (`t_dictionary` \*d, `t_symbol` \*key, `t_atom` \*value, `t_atom` \*def)  
*Retrieve a `t_atom`\* from the dictionary.*
- `t_max_err` `dictionary_getdefstring` (`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 (char *fmt,...)`  
*Create a new dictionary populated with values using a combination of attribute and sprintf syntax.*
- `t_max_err dictionary_read (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, char *filename, short path)`  
*Serialize the specified `t_dictionary` object to a JSON file.*
- `void postdictionary (t_object *d)`  
*Print the contents of a dictionary to the Max window.*

### 26.8.1 Detailed Description

In Max 5, we have a new "dictionary" object which can be used for object prototypes, object serialization, object constructors, and 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 entites, 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).

### 26.8.2 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 recipies for using dictionaries to accomplish common tasks.

#### 26.8.2.1 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);
}
```

### 26.8.2.2 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()` whose name makes it clear that the dictionary will be deleted after calling the function.

### 26.8.2.3 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

## 26.8.3 Function Documentation

### 26.8.3.1 `t_max_err dictionary_appendatom (t_dictionary * d, t_symbol * key, t_atom * value)`

Add a `t_atom*` value to the dictionary.

Parameters:

*d* The dictionary instance.

*key* 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.

*value* The new value to append to the dictionary.

Returns:

A Max error code.

### 26.8.3.2 `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:**

*d* The dictionary instance.

*key* 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.

*value* The new value to append to the dictionary.

**Returns:**

A Max error code.

**26.8.3.3 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:**

*d* The dictionary instance.

*key* 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.

*argc* The number of atoms to append to the dictionary.

*argv* The address of the first atom in the array to append to the dictionary.

**Returns:**

A Max error code.

**26.8.3.4 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:**

*d* The dictionary instance.

*key* 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.

*value* The new value to append to the dictionary.

**Returns:**

A Max error code.

**26.8.3.5 t\_max\_err dictionary\_appendfloat (t\_dictionary \* *d*, t\_symbol \* *key*, double *value*)**

Add a double-precision float value to the dictionary.

**Parameters:**

*d* The dictionary instance.

*key* 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.

*value* The new value to append to the dictionary.

**Returns:**

A Max error code.

**26.8.3.6 t\_max\_err dictionary\_appendlong (t\_dictionary \* *d*, t\_symbol \* *key*, long *value*)**

Add a long integer value to the dictionary.

**Parameters:**

*d* The dictionary instance.

*key* 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.

*value* The new value to append to the dictionary.

**Returns:**

A Max error code.

**26.8.3.7 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:**

*d* The dictionary instance.

*key* 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.

*value* The new value to append to the dictionary.

**Returns:**

A Max error code.

**26.8.3.8 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\\_symbol](#) 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:**

*d* The dictionary instance.

*key* 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.

*value* The new value to append to the dictionary.

**Returns:**

A Max error code.

**26.8.3.9 t\_max\_err dictionary\_appendsym (t\_dictionary \* *d*, t\_symbol \* *key*, t\_symbol \* *value*)**

Add a [t\\_symbol](#)\* value to the dictionary.

**Parameters:**

*d* The dictionary instance.

*key* 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.

*value* The new value to append to the dictionary.

**Returns:**

A Max error code.

**26.8.3.10 t\_max\_err dictionary\_chuckentry (t\_dictionary \* *d*, t\_symbol \* *key*)**

Remove a value from the dictionary without freeing it.

**Parameters:**

*d* The dictionary instance.

*key* The key associated with the value to delete.

**Returns:**

A max error code.

**See also:**

[dictionary\\_deleteentry\(\)](#)



**26.8.3.11 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:**

*d* The dictionary instance.

**Returns:**

A max error code.

**See also:**

[dictionary\\_getkeys\(\)](#)  
[dictionary\\_chuckentry\(\)](#)  
[dictionary\\_deleteentry\(\)](#)

**26.8.3.12 t\_max\_err dictionary\_copyatoms (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 [system\\_freeptr\(\)](#).

**Parameters:**

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*argc* The address of a variable to hold the number of atoms in the array.

*argv* 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 [dictionary\\_copyatoms\(\)](#).

**Returns:**

A Max error code.

**See also:**

[dictionary\\_getatoms\(\)](#)

**26.8.3.13 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 [system\\_freeptr\(\)](#). If the named key doesn't exist, then copy a default array of atoms, specified as a [t\\_atomarray\\*](#).

**Parameters:**

*d* The dictionary instance.

- key* The key associated with the value to lookup.
- argc* The address of a variable to hold the number of atoms in the array.
- argv* 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 [dictionary\\_copyatoms\(\)](#).
- def* The default values specified as an instance of the [t\\_atomarray](#) object.

**Returns:**

A Max error code.

**See also:**

[dictionary\\_getdefatoms\(\)](#)  
[dictionary\\_copyatoms\(\)](#)

**26.8.3.14 t\_max\_err dictionary\_copyentries (t\_dictionary \* src, t\_dictionary \* dst, t\_symbol \*\* keys)**

Copy specified entries from one dictionary to another.

**Parameters:**

- src* The source dictionary from which to copy entries.
- dst* The destination dictionary to which the entries will be copied.
- keys* The address of the first of an array of [t\\_symbol\\*](#) that specifies which keys to copy.

**Returns:**

A Max error code.

**See also:**

[dictionary\\_copyunique\(\)](#)

**26.8.3.15 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:**

- d* A dictionary instance. This will be the destination for any values that are copied.
- copyfrom* A dictionary instance from which we will copy any values with unique keys.

**Returns:**

A Max error code.

**See also:**

[dictionary\\_copyentries\(\)](#)

**26.8.3.16 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:**

*d* The dictionary instance.

*key* The key associated with the value to delete.

**Returns:**

A max error code.

**See also:**

[dictionary\\_chuckentry\(\)](#)  
[dictionary\\_clear\(\)](#)

**26.8.3.17 t\_max\_err dictionary\_dump (t\_dictionary \* *d*, long *recurse*, long *console*)**

Print the contents of a dictionary to the Max window.

**Parameters:**

*d* The dictionary instance.

*recurse* If non-zero, the dictionary will be recursively unravelled to the Max window. Otherwise it will only print the top level.

*console* 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.

**26.8.3.18 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\(\)](#)

**26.8.3.19 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 [t\\_atom](#) to which the value will be copied.

**See also:**

[dictionary\\_entry\\_getkey\(\)](#)

[dictionary\\_funall\(\)](#)

**26.8.3.20 long dictionary\_entryisatomarray (t\_dictionary \* *d*, t\_symbol \* *key*)**

Test a key to set if the data stored with that key contains a [t\\_atomarray](#) object.

**Parameters:**

*d* The dictionary instance.

*key* The key associated with the value to test.

**Returns:**

Returns true if the key contains a [t\\_atomarray](#), otherwise returns false.

**26.8.3.21 long dictionary\_entryisdictionary (t\_dictionary \* *d*, t\_symbol \* *key*)**

Test a key to set if the data stored with that key contains a [t\\_dictionary](#) object.

**Parameters:**

*d* The dictionary instance.

*key* The key associated with the value to test.

**Returns:**

Returns true if the key contains a [t\\_dictionary](#), otherwise returns false.

**26.8.3.22 long dictionary\_entryisstring (t\_dictionary \* *d*, t\_symbol \* *key*)**

Test a key to set if the data stored with that key contains a [t\\_string](#) object.

**Parameters:**

*d* The dictionary instance.

*key* The key associated with the value to test.

**Returns:**

Returns true if the key contains a [t\\_string](#), otherwise returns false.

**26.8.3.23 void dictionary\_freekeys (t\_dictionary \* *d*, long *numkeys*, t\_symbol \*\* *keys*)**

Free memory allocated by the [dictionary\\_getkeys\(\)](#) method.

**Parameters:**

*d* The dictionary instance.

*numkeys* The address of a long where the number of keys retrieved will be set.

*keys* The address of the first of an array [t\\_symbol](#) pointers where the retrieved keys will be set.

**Returns:**

A max error code.

**See also:**

[dictionary\\_getkeys\(\)](#)

**26.8.3.24 void dictionary\_funall (t\_dictionary \* *d*, method *fun*, void \* *arg*)**

Call the specified function for every entry in the dictionary.

**Parameters:**

*d* The dictionary 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.

**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\(\)](#)

**26.8.3.25 t\_max\_err dictionary\_getatom (t\_dictionary \* *d*, t\_symbol \* *key*, t\_atom \* *value*)**

Copy a [t\\_atom](#) from the dictionary.

**Parameters:**

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*value* The address of variable to hold the value associated with the key.

**Returns:**

A Max error code.

**26.8.3.26** `t_max_err dictionary_getatomarray (t_dictionary * d, t_symbol * key, t_object ** value)`

Retrieve a [t\\_atomarray](#) pointer from the dictionary.

**Parameters:**

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*value* The address of variable to hold the value associated with the key.

**Returns:**

A Max error code.

**26.8.3.27** `t_max_err dictionary_getatoms (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:**

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*argc* The address of a variable to hold the number of atoms in the array.

*argv* 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\(\)](#)

**26.8.3.28** `t_max_err dictionary_getdefatom (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:**

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*value* The address of variable to hold the value associated with the key.

*def* The default value to return in the absence of the key existing in the dictionary.

#### Returns:

A Max error code.

#### See also:

[dictionary\\_getatom\(\)](#)

#### 26.8.3.29 `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:

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*argc* The address of a variable to hold the number of atoms in the array.

*argv* The address of a variable to hold a pointer to the first atom in the array.

*def* The default values specified as an instance of the [t\\_atomarray](#) object.

#### Returns:

A Max error code.

#### See also:

[dictionary\\_getatoms\(\)](#)

[dictionary\\_copydefatoms\(\)](#)

#### 26.8.3.30 `t_max_err dictionary_getdeffloat (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:

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*value* The address of variable to hold the value associated with the key.

*def* The default value to return in the absence of the key existing in the dictionary.

#### Returns:

A Max error code.

#### See also:

[dictionary\\_getfloat\(\)](#)

### 26.8.3.31 `t_max_err dictionary_getdeflong (t_dictionary * d, t_symbol * key, long * value, long def)`

Retrieve a long integer from the dictionary. If the named key doesn't exist, then return a specified default value.

#### Parameters:

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*value* The address of variable to hold the value associated with the key.

*def* The default value to return in the absence of the key existing in the dictionary.

#### Returns:

A Max error code.

#### See also:

[dictionary\\_getlong\(\)](#)

### 26.8.3.32 `t_max_err dictionary_getdefstring (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:

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*value* The address of variable to hold the value associated with the key.

*def* The default value to return in the absence of the key existing in the dictionary.

#### Returns:

A Max error code.

#### See also:

[dictionary\\_getstring\(\)](#)

### 26.8.3.33 `t_max_err dictionary_getdefsym (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:

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*value* The address of variable to hold the value associated with the key.



*def* The default value to return in the absence of the key existing in the dictionary.

**Returns:**

A Max error code.

**See also:**

[dictionary\\_getsym\(\)](#)

**26.8.3.34 t\_max\_err dictionary\_getdictionary (t\_dictionary \* *d*, t\_symbol \* *key*, t\_object \*\* *value*)**

Retrieve a [t\\_dictionary](#) pointer from the dictionary.

**Parameters:**

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*value* The address of variable to hold the value associated with the key.

**Returns:**

A Max error code.

**26.8.3.35 long dictionary\_getentrycount (t\_dictionary \* *d*)**

Return the number of keys in a dictionary.

**Parameters:**

*d* The dictionary instance.

**Returns:**

The number of keys in the dictionary.

**26.8.3.36 t\_max\_err dictionary\_getfloat (t\_dictionary \* *d*, t\_symbol \* *key*, double \* *value*)**

Retrieve a double-precision float from the dictionary.

**Parameters:**

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*value* The address of variable to hold the value associated with the key.

**Returns:**

A Max error code.

**26.8.3.37 t\_max\_err dictionary\_getkeys (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:**

*d* The dictionary instance.

*numkeys* The address of a long where the number of keys retrieved will be set.

*keys* The address of the first of an array [t\\_symbol](#) 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\(\)](#)

**26.8.3.38 t\_max\_err dictionary\_getlong (t\_dictionary \* *d*, t\_symbol \* *key*, long \* *value*)**

Retrieve a long integer from the dictionary.

**Parameters:**

*d* The dictionary instance.

*key* The key associated with the value to lookup.

*value* The address of variable to hold the value associated with the key.

**Returns:**

A Max error code.

**26.8.3.39 t\_max\_err dictionary\_getobject (t\_dictionary \* *d*, t\_symbol \* *key*, t\_object \*\* *value*)**

Retrieve a [t\\_object](#) pointer from the dictionary.

**Parameters:**

- d* The dictionary instance.
- key* The key associated with the value to lookup.
- value* The address of variable to hold the value associated with the key.

**Returns:**

A Max error code.

**26.8.3.40 t\_max\_err dictionary\_getstring (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:**

- d* The dictionary instance.
- key* The key associated with the value to lookup.
- value* The address of variable to hold the value associated with the key.

**Returns:**

A Max error code.

**26.8.3.41 t\_max\_err dictionary\_getsym (t\_dictionary \* *d*, t\_symbol \* *key*, t\_symbol \*\* *value*)**

Retrieve a [t\\_symbol\\*](#) from the dictionary.

**Parameters:**

- d* The dictionary instance.
- key* The key associated with the value to lookup.
- value* The address of variable to hold the value associated with the key.

**Returns:**

A Max error code.

**26.8.3.42 long dictionary\_hasentry (t\_dictionary \* *d*, t\_symbol \* *key*)**

Test a key to see if it exists in the dictionary.

**Parameters:**

- d* The dictionary instance.
- key* The key associated with the value to test.

**Returns:**

Returns true if the key exists, otherwise returns false.

### 26.8.3.43 `t_dictionary* dictionary_new ()`

Create a new linklist object. You can free the linklist 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\(\)](#)

### 26.8.3.44 `t_max_err dictionary_read (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:

*filename* The name of the file.

*path* The path of the file.

*d* The address of a [t\\_dictionary](#) pointer that will be set to the newly created dictionary.

#### Returns:

A Max error code

### 26.8.3.45 `t_dictionary* dictionary_sprintf (char *fmt, ...)`

Create a new dictionary populated with values using a combination of attribute and sprintf syntax.

#### Parameters:

*fmt* 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\(\)](#)

**26.8.3.46 t\_max\_err dictionary\_write (t\_dictionary \* *d*, char \**filename*, short *path*)**

Serialize the specified [t\\_dictionary](#) object to a JSON file.

**Parameters:**

*d* The dictionary to serialize into JSON format and write to disk.  
*filename* The name of the file to write.  
*path* The path to which the file should be written.

**Returns:**

A Max error code.

**26.8.3.47 void postdictionary (t\_object \* *d*)**

Print the contents of a dictionary to the Max window.

**Parameters:**

*d* A pointer to a dictionary object.

## 26.9 Hash Table

Max's hashtable object implements a hash table ([http://en.wikipedia.org/wiki/Hash\\_table](http://en.wikipedia.org/wiki/Hash_table)).

Collaboration diagram for Hash Table:



### Data Structures

- struct [t\\_hashtab\\_entry](#)  
*A hashtable entry.*
- struct [t\\_hashtab](#)  
*The hashtable object.*

### Defines

- #define [HASH\\_DEFSLOTS](#) 57  
*Default number of slots in the hash table.*

### Functions

- [t\\_hashtab \\* hashtable\\_new](#) (long slotcount)  
*Create a new hashtable object.*
- [t\\_max\\_err hashtable\\_store](#) (t\_hashtab \*x, t\_symbol \*key, t\_object \*val)  
*Store an item in a hashtable with an associated key.*
- [t\\_max\\_err hashtable\\_store\\_safe](#) (t\_hashtab \*x, t\_symbol \*key, t\_object \*val)  
*Store an item in a hashtable with an associated key.*
- [t\\_max\\_err hashtable\\_storeflags](#) (t\_hashtab \*x, t\_symbol \*key, t\_object \*val, long flags)  
*Store an item in a hashtable with an associated key and also flags that define the behavior of the item.*
- [t\\_max\\_err hashtable\\_lookup](#) (t\_hashtab \*x, t\_symbol \*key, t\_object \*\*val)  
*Return an item stored in a hashtable with the specified key.*
- [t\\_max\\_err hashtable\\_lookupflags](#) (t\_hashtab \*x, t\_symbol \*key, t\_object \*\*val, long \*flags)  
*Return an item stored in a hashtable with the specified key, also returning the items flags.*
- [t\\_max\\_err hashtable\\_delete](#) (t\_hashtab \*x, t\_symbol \*key)  
*Remove an item from a hashtable associated with the specified key and free it.*
- [t\\_max\\_err hashtable\\_clear](#) (t\_hashtab \*x)

*Delete all items stored in a hashtable.*

- `t_max_err hashtable_chuckkey (t_hashtab *x, t_symbol *key)`  
*Remove an item from a hashtable associated with a given key.*
- `t_max_err hashtable_chuck (t_hashtab *x)`  
*Free a hashtable, but don't free the items it contains.*
- `t_max_err hashtable_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 hashtable_methodall (t_hashtab *x, t_symbol *s,...)`  
*Call the named message on every object in the hashtable.*
- `t_max_err hashtable_funall (t_hashtab *x, method fun, void *arg)`  
*Call the specified function for every item in the hashtable.*
- `long hashtable_getsize (t_hashtab *x)`  
*Return the number of items stored in a hashtable.*
- `void hashtable_print (t_hashtab *x)`  
*Post a hashtable's statistics to the Max window.*
- `void hashtable_readonly (t_hashtab *x, long readonly)`  
*Set the hashtable's readonly bit.*
- `void hashtable_flags (t_hashtab *x, long flags)`  
*Set the hashtable's datastore flags.*
- `long hashtable_getflags (t_hashtab *x)`  
*Get the hashtable's datastore flags.*
- `t_max_err hashtable_keyflags (t_hashtab *x, t_symbol *key, long flags)`  
*Change the flags for an item stored in the hashtable with a given key.*
- `long hashtable_getkeyflags (t_hashtab *x, t_symbol *key)`  
*Retrieve the flags for an item stored in the hashtable with a given key.*
- `t_max_err hashtable_getkeys (t_hashtab *x, long *kc, t_symbol ***kv)`  
*Retrieve all of the keys stored in a hashtable.*

### 26.9.1 Detailed Description

Max's hashtable object implements a hash table ([http://en.wikipedia.org/wiki/Hash\\_table](http://en.wikipedia.org/wiki/Hash_table)). Any type of value may be stored in the table, indexed using a `t_symbol` as the unique key.

**See also:**

[Linked List](#)

## 26.9.2 Define Documentation

### 26.9.2.1 #define HASH\_DEFSLOTS 57

Default number of slots in the hash table. Creating a hashtable using [hashtab\\_new\(\)](#) with an argument of 0 will use the default number of slots. Primes typically work well for the number of slots.

Definition at line 17 of file `ext_hashtab.h`.

## 26.9.3 Function Documentation

### 26.9.3.1 `t_max_err hashtab_chuck (t_hashtab * x)`

Free a hashtable, but don't free the items it contains. The hashtable can contain a variety of different types of data. By default, the hashtable assumes that all items are max objects with a valid [t\\_object](#) header.

You can alter the hashtable's notion of what it contains by using the [hashtab\\_flags\(\)](#) method.

When you free the hashtable by calling [object\\_free\(\)](#) it then tries to free all of the items it contains. If the hashtable 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 hashtable object to be freed.

#### Returns:

A max error code.

#### See also:

[object\\_free](#)

### 26.9.3.2 `t_max_err hashtab_chuckkey (t_hashtab * x, t_symbol * key)`

Remove an item from a hashtable associated with a given key. You are responsible for freeing any memory associated with the item that is removed from the hashtable.

#### Parameters:

*x* The hashtable instance.

*key* The key of the item to delete.

#### Returns:

A Max error code.

#### See also:

[hashtab\\_delete](#)



### 26.9.3.3 `t_max_err hashtable_clear (t_hashtab * x)`

Delete all items stored in a hashtable. This is the equivalent of calling [hashtab\\_delete\(\)](#) on every item in a hashtable.

**Returns:**

A max error code.

**See also:**

[hashtab\\_flags\(\)](#)  
[hashtab\\_delete\(\)](#)

### 26.9.3.4 `t_max_err hashtable_delete (t_hashtab * x, t_symbol * key)`

Remove an item from a hashtable associated with the specified key and free it. The hashtable can contain a variety of different types of data. By default, the hashtable 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 hashtable's notion of what it contains by using the [hashtab\\_flags\(\)](#) method.

If you wish to remove an item from the hashtable and free it yourself, then you should use [hashtab\\_chuckkey\(\)](#).

**Parameters:**

*x* The hashtable instance.

*key* The key of the item to delete.

**Returns:**

A Max error code.

**See also:**

[hashtab\\_chuckkey\(\)](#)  
[hashtab\\_clear\(\)](#)  
[hashtab\\_flags\(\)](#)

### 26.9.3.5 `t_max_err hashtable_findfirst (t_hashtab * x, void ** o, long cmpfnvoid *, void *, void * cmpdata)`

Search the hash table for the first item meeting defined criteria. The items in the hashtable are iteratively processed, calling a specified comparison function on each until the comparison function returns true.

**Parameters:**

*x* The hashtable instance.

*o* The address to pointer that will be set with the matching item.

*cmpfn* The function used to determine a match in the list.

*cmpdata* An argument to be passed to the [t\\_cmpfn](#). This will be passed as the second of the two args to the [t\\_cmpfn](#). The first arg will be the hashtable item at each iteration in the list.

**Returns:**

A max error code.

**See also:**

[linklist\\_findfirst\(\)](#)  
[t\\_cmpfn](#)

**26.9.3.6 void hashtable\_flags (t\_hashtab \* *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 [hashtab\\_clear\(\)](#), [hashtab\\_delete\(\)](#), or when freeing the hashtable itself.

**Parameters:**

*x* The hashtable instance.

*flags* A valid value from the [e\\_max\\_datastore\\_flags](#). The default is [OBJ\\_FLAG\\_OBJ](#).

**26.9.3.7 t\_max\_err hashtable\_funall (t\_hashtab \* *x*, method *fun*, void \* *arg*)**

Call the specified function for every item in the hashtable.

**Parameters:**

*x* The hashtable 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 [hashtab\\_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_hashtab_entry *e, void *myArg)
{
    if (e->key && e->value) {
        // do something with e->key, e->value, and myArg here as appropriate
    }
}
```

**26.9.3.8 long hashtable\_getflags (t\_hashtab \* *x*)**

Get the hashtable's datastore flags.

**Parameters:**

*x* The hashtable instance.

**Returns:**

The current state of the hashtable flags as enumerated in [e\\_max\\_datastore\\_flags](#).

### 26.9.3.9 long hashtable\_getkeyflags (t\_hashtab \* *x*, t\_symbol \* *key*)

Retrieve the flags for an item stored in the hashtable with a given key.

**Parameters:**

- x* The hashtable instance.
- key* The key in the hashtable whose flags will be returned.

**Returns:**

The flags for the given key.

**See also:**

hashtable\_store\_flags()

### 26.9.3.10 t\_max\_err hashtable\_getkeys (t\_hashtab \* *x*, long \* *kc*, t\_symbol \*\*\* *kv*)

Retrieve all of the keys stored in a hashtable. If the *kc* and *kv* parameters are properly initialized to zero, then [hashtable\\_getkeys\(\)](#) will allocate memory for the keys it returns. You are then responsible for freeing this memory using [systemem\\_freeptr\(\)](#).

**Parameters:**

- x* The hashtable 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 [t\\_symbol](#) 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 hashtable in order to iterate through each item stored in the hashtable.

```
t_symbol    **keys = NULL;
long        numKeys = 0;
long        i;
t_object    *anItem;

hashtable_getkeys(aHashtab, &numKeys, &keys);
for(i=0; i<numKeys; i++){
    hashtable_lookup(aHashtab, keys[i], &anItem);
    // Do something with anItem here...
}
if(keys)
    systemem_freeptr(keys);
```

### 26.9.3.11 long hashtable\_getsize (t\_hashtab \* *x*)

Return the number of items stored in a hashtable.

**Parameters:**

*x* The hashtable instance.

**Returns:**

The number of items in the hash table.

**26.9.3.12 t\_max\_err hashtable\_keyflags (t\_hashtab \* *x*, t\_symbol \* *key*, long *flags*)**

Change the flags for an item stored in the hashtable with a given key.

**Parameters:**

*x* The hashtable instance.

*key* The key in the hashtable whose flags will be changed.

*flags* One of the values listed in [e\\_max\\_datastore\\_flags](#).

**Returns:**

A Max error code.

**See also:**

[hashtab\\_store\\_flags\(\)](#)

**26.9.3.13 t\_max\_err hashtable\_lookup (t\_hashtab \* *x*, t\_symbol \* *key*, t\_object \*\* *val*)**

Return an item stored in a hashtable with the specified key.

**Parameters:**

*x* The hashtable instance.

*key* The key in the hashtable to fetch.

*val* The address of a pointer to which the fetched value will be assigned.

**Returns:**

A Max error code.

**See also:**

[hashtab\\_store\(\)](#)

**26.9.3.14 t\_max\_err hashtable\_lookupflags (t\_hashtab \* *x*, t\_symbol \* *key*, t\_object \*\* *val*, long \* *flags*)**

Return an item stored in a hashtable with the specified key, also returning the items flags.

**Parameters:**

*x* The hashtable instance.

*key* The key in the hashtable to fetch.

*val* The address of a pointer to which the fetched value will be assigned.

*flags* 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\(\)](#)

**26.9.3.15 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:**

*x* The hashtab instance.  
*s* 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.

**26.9.3.16 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:**

*slotcount* 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\(\)](#)

**26.9.3.17 void hashtable\_print (t\_hashtab \* x)**

Post a hashtable's statistics to the Max window.

**Parameters:**

*x* The hashtable instance.

**26.9.3.18 void hashtable\_readonly (t\_hashtab \* 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 theadsafety 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:**

*x* The hashtable instance.

*readonly* A 1 or 0 for setting the readonly bit.

**26.9.3.19 t\_max\_err hashtable\_store (t\_hashtab \* x, t\_symbol \* key, t\_object \* val)**

Store an item in a hashtable with an associated key.

**Parameters:**

*x* The hashtable instance.

*key* The key in the hashtable with which to associate the value.

*val* The value to store.

**Returns:**

A Max error code.

**See also:**

[hashtab\\_lookup\(\)](#)

**26.9.3.20 t\_max\_err hashtable\_store\_safe (t\_hashtab \* x, t\_symbol \* key, t\_object \* val)**

Store an item in a hashtable 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 [systemem\\_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:**

*x* The hashtable instance.

*key* The key in the hashtable with which to associate the value.

*val* The value to store.

**Returns:**

A Max error code.

**See also:**

[hashtab\\_store\(\)](#)

**26.9.3.21 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:**

*x* The hashtab instance.

*key* The key in the hashtab with which to associate the value.

*val* The value to store.

*flags* One of the values listed in [e\\_max\\_datastore\\_flags](#).

**Returns:**

A Max error code.

**See also:**

[hashtab\\_store\(\)](#)

## 26.10 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.*

### 26.10.1 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 hashtable 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 hashtable are updated with new indices.

### 26.10.2 Function Documentation

#### 26.10.2.1 void `indexmap_append` (`t_indexmap *x`, void \* *data*)

Add an item to an indexmap.

**Parameters:**

- x* The indexmap instance.
- data* The item to add.

#### 26.10.2.2 void `indexmap_clear` (`t_indexmap *x`)

Delete all items in an indexmap.

**Parameters:**

- x* The indexmap instance.

#### 26.10.2.3 void\* `indexmap_datafromindex` (`t_indexmap *x`, long *index*)

Get an item from an indexmap by index.

**Parameters:**

- x* The indexmap instance.
- index* The index from which to fetch a stored item.

**Returns:**

- The item stored at the specified index.

**26.10.2.4 t\_max\_err indexmap\_delete (t\_indexmap \* *x*, void \* *data*)**

Delete a specified item from an indexmap.

**Parameters:**

- x* The indexmap instance.
- data* The item pointer to remove from the indexmap.

**Returns:**

A Max error code.

**26.10.2.5 t\_max\_err indexmap\_delete\_index (t\_indexmap \* *x*, long *index*)**

Delete an item from the indexmap by index.

**Parameters:**

- x* The indexmap instance.
- index* The index of the item to remove from the indexmap.

**Returns:**

A Max error code.

**26.10.2.6 t\_max\_err indexmap\_delete\_index\_multi (t\_indexmap \* *x*, long *count*, long \* *indices*)**

Delete multiple items from an indexmap by index.

**Parameters:**

- x* The indexmap instance.
- count* The number of items to remove from the indexmap.
- indices* The address of the first of an array of index numbers to remove the indexmap.

**Returns:**

A Max error code.

**26.10.2.7 t\_max\_err indexmap\_delete\_multi (t\_indexmap \* *x*, long *count*, void \*\* *pdata*)**

Delete multiple specified items from an indexmap.

**Parameters:**

- x* The indexmap instance.
- count* The number of items to remove from the indexmap.
- pdata* The address of the first of an array of item pointers to remove from the indexmap.

**Returns:**

A Max error code.

**26.10.2.8 long indexmap\_getsize (t\_indexmap \* *x*)**

Return the number of items in an indexmap.

**Parameters:**

*x* The indexmap instance.

**Returns:**

The number of items in the indexmap.

**26.10.2.9 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:**

*x* The indexmap instance.

*data* The item whose index you wish to look up.

*index* The address of a variable to hold the retrieved index.

**Returns:**

A Max error code.

**26.10.2.10 t\_max\_err indexmap\_move (t\_indexmap \* *x*, void \* *data*, long *newindex*)**

Move an item to a different position in an indexmap.

**Parameters:**

*x* The indexmap instance.

*data* The item in the indexmap to move.

*newindex* The new index to which to move the item.

**Returns:**

A Max error code.

**26.10.2.11 t\_indexmap\* indexmap\_new (void)**

Create a new indexmap object.

**Returns:**

Pointer to the new indexmap object.

**26.10.2.12 void indexmap\_sort (t\_indexmap \* *x*, t\_cmpfn *fn*)**

Sort the items in an indexmap. Item are sorted using a [t\\_cmpfn](#) function that is passed in as an argument.

**Parameters:**

*x* The indexmap instance.

*fn* The function used to sort the list.

**See also:**

[linklist\\_sort\(\)](#)

## 26.11 Linked List

Max's linklist object implements a doubly-linked-list ([http://en.wikipedia.org/wiki/Linked\\_list](http://en.wikipedia.org/wiki/Linked_list)) together with a high-level interface for manipulating and accessing values in the list.

Collaboration diagram for Linked List:



### Data Structures

- struct `t_llelem`  
*A linklist element.*
- struct `t_linklist`  
*The linklist object.*

### Functions

- `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.*
- `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.*
- `long linklist_objptr2index (t_linklist *x, void *p)`  
*Return an item's index, given the item itself.*
- `long linklist_append (t_linklist *x, void *o)`  
*Add an item to the end of the list.*
- `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.*
- `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.*
- `void linklist_chuckobject (t_linklist *x, void *o)`  
*Remove the specified item from the list.*
- `void linklist_clear (t_linklist *x)`  
*Remove and free all items in the list.*
- `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.*
- `long linklist_findfirst (t_linklist *x, void **o, long cmpfn(void *, void *), void *cmpdata)`  
*Search the linked list for the first item meeting defined criteria.*
- `void linklist_findall (t_linklist *x, t_linklist **out, long cmpfn(void *, void *), void *cmpdata)`  
*Search the linked list for all items meeting defined criteria.*
- `void linklist_methodall (t_linklist *x, t_symbol *s,...)`  
*Call the named message on every object in the linklist.*
- `void * linklist_methodindex (t_linklist *x, long i, t_symbol *s,...)`  
*Call the named message on an object specified by index.*
- `void linklist_sort (t_linklist *x, long cmpfn(void *, void *))`  
*Sort the linked list.*

- void `linklist_funall` (`t_linklist` \*x, `method` fun, void \*arg)  
*Call the specified function for every item in the linklist.*
- 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.*
- 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.*

### 26.11.1 Detailed Description

Max's linklist object implements a doubly-linked-list ([http://en.wikipedia.org/wiki/Linked\\_list](http://en.wikipedia.org/wiki/Linked_list)) together with a high-level interface for manipulating and accessing values in the list.

### 26.11.2 Function Documentation

#### 26.11.2.1 long `linklist_append` (`t_linklist` \*x, void \*o)

Add an item to the end of the list.

##### Parameters:

- x* The linklist instance.
- o* The item pointer to append to the linked-list.

**Returns:**

The index of the item in the linklist.

**26.11.2.2 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:**

*x* The linklist object to be freed.

**See also:**

[object\\_free](#)

**26.11.2.3 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:**

*x* The linklist instance.

*index* 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](#)

**26.11.2.4 void 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:**

*x* The linklist instance.

*o* The pointer to the item to remove.

**Returns:**

Returns [MAX\\_ERR\\_NONE](#) on successful removal, otherwise returns [MAX\\_ERR\\_GENERIC](#)



See also:

[linklist\\_deleteindex](#)  
[linklist\\_chuckindex](#)

#### 26.11.2.5 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:

*x* The linklist instance.

#### 26.11.2.6 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:

*x* The linklist instance.

*index* The index of the item to delete.

Returns:

Returns [MAX\\_ERR\\_NONE](#) on successful deletion, otherwise returns [MAX\\_ERR\\_GENERIC](#)

See also:

[linklist\\_chuckindex](#)  
[linklist\\_chuckobject](#)

#### 26.11.2.7 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:

*x* The linklist instance.

*out* The address to a [t\\_linklist](#) pointer. You should initialize the pointer to NULL before calling [linklist\\_findall\(\)](#). A new linklist will be created internally by [linklist\\_findall\(\)](#) and returned here.

*cmpfn* The function used to determine a match in the list.

*cmpdata* An argument to be passed to the [t\\_cmpfn](#). This will be passed as the second of the two args to the [t\\_cmpfn](#). 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](#)

**26.11.2.8 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:**

*x* The linklist instance.

*o* The address to pointer that will be set with the matching item.

*cmpfn* The function used to determine a match in the list.

*cmpdata* An argument to be passed to the [t\\_cmpfn](#). This will be passed as the second of the two args to the [t\\_cmpfn](#). 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](#)

**26.11.2.9 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:**

*x* The linklist instance.

*flags* A valid value from the [e\\_max\\_datastore\\_flags](#). The default is [OBJ\\_FLAG\\_OBJ](#).

**26.11.2.10 void linklist\_funall (t\_linklist \* x, method fun, void \* arg)**

Call the specified function for every item in the linklist.

**Parameters:**

*x* The linklist 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.

**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
}
```

**26.11.2.11 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:**

*x* The linklist 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.

**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 *myListItemSymbol, void *myArg)
{
    // this function is called by a linklist that contains symbols for its it
    ems
    if(myListItemSymbol == gensym("")){
```

```

        error("empty symbol -- aborting linklist traversal")
        return 1;
    }
    else{
        // do something with the symbol
        return 0;
    }
}

```

#### 26.11.2.12 void\* linklist\_funindex (t\_linklist \*x, long i, method fun, void \*arg)

Call the specified function for an item specified by index.

##### Parameters:

*x* The linklist instance.

*i* The index of the item to which to send the message.

*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.

##### 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
}

```

#### 26.11.2.13 long linklist\_getflags (t\_linklist \*x)

Get the linklist's datastore flags.

##### Parameters:

*x* The linklist instance.

##### Returns:

The current state of the linklist flags as enumerated in [e\\_max\\_datastore\\_flags](#).

#### 26.11.2.14 void\* linklist\_getindex (t\_linklist \*x, long index)

Return the item stored in a linklist at a specified index.

##### Parameters:

*x* The linklist instance.

*index* 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

**26.11.2.15** `long linklist_getsize (t_linklist * x)`

Return the number of items in a linklist object.

**Parameters:**

*x* The linklist instance.

**Returns:**

The number of items in the linklist object.

**26.11.2.16** `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:**

*x* The linklist instance.

*o* The item pointer to insert.

*cmpfn* 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.

**26.11.2.17** `t_llelem* linklist_insertafterobjptr (t_linklist * x, void * o, void * objptr)`

Insert an item into the list after another specified item.

**Parameters:**

*x* The linklist instance.

*o* The item pointer to insert.

*objptr* The item pointer after which to insert in the list.

**Returns:**

An opaque linklist element.

**26.11.2.18** `t_llelem* linklist_insertbeforeobjptr (t_linklist * x, void * o, void * objptr)`

Insert an item into the list before another specified item.

**Parameters:**

*x* The linklist instance.

*o* The item pointer to insert.

*objptr* The item pointer before which to insert in the list.

**Returns:**

An opaque linklist element.

**26.11.2.19 long linklist\_insertindex (t\_linklist \* *x*, void \* *o*, long *index*)**

Insert an item into the list at the specified index.

**Parameters:**

*x* The linklist instance.

*o* The item pointer to insert.

*index* 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.

**26.11.2.20 void\* linklist\_last (t\_linklist \* *x*, void \*\* *item*)**

Return the last item (the tail) in the linked-list.

**Parameters:**

*x* The linklist instance.

*item* The address of pointer in which to store the last item in the linked-list.

**Returns:**

always returns NULL

**26.11.2.21 long linklist\_makearray (t\_linklist \* *x*, void \*\* *a*, long *max*)**

Retrieve linklist items as an array of pointers.

**Parameters:**

*x* The linklist instance.

*a* The address of the first pointer in the array to fill.

*max* The number of pointers in the array.

**Returns:**

The number of items from the list actually returned in the array.

**26.11.2.22 long linklist\_match (void \* *a*, void \* *b*)**

A linklist comparison method that determines if two item pointers are equal.

**Parameters:**

*a* The first item to compare.

*b* The second item to compare.

**Returns:**

Returns 1 if the items are equal, otherwise 0.

**See also:**

[t\\_cmpfn](#)

**26.11.2.23 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:**

- x* The linklist instance.
- s* 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.

**26.11.2.24 void\* linklist\_methodindex (t\_linklist \* *x*, 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:**

- x* The linklist instance.
- i* The index of the item to which to send the message.
- s* 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.

**26.11.2.25 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:**

- x* The linklist instance.
- o* The item pointer to insert.

*objptr* The item pointer after which to move *o* in the list.

**Returns:**

An opaque linklist element.

**26.11.2.26** `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:**

*x* The linklist instance.

*o* The item pointer to insert.

*objptr* The item pointer before which to move *o* in the list.

**Returns:**

An opaque linklist element.

**26.11.2.27** `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\(\)](#)

**26.11.2.28** `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:**

*x* The linklist instance.

*p* An item in the list.

*next* The address of a pointer to set with the next item in the list.

**26.11.2.29** `long linklist_objptr2index (t_linklist * x, void * p)`

Return an item's index, given the item itself.

**Parameters:**

*x* The linklist instance.



*p* 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.

**26.11.2.30 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:**

*x* The linklist instance.

*p* An item in the list.

*prev* The address of a pointer to set with the previous item in the list.

**26.11.2.31 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 theadsafety 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:**

*x* The linklist instance.

*readonly* A 1 or 0 for setting the readonly bit.

**26.11.2.32 void linklist\_reverse (t\_linklist \* *x*)**

Reverse the order of items in the linked-list.

**Parameters:**

*x* The linklist instance.

**26.11.2.33 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.

### 26.11.2.34 void linklist\_shuffle (t\_linklist \*x)

Randomize the order of items in the linked-list.

#### Parameters:

*x* The linklist instance.

### 26.11.2.35 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);
}
```

### 26.11.2.36 void\* linklist\_substitute (t\_linklist \*x, void \*p, void \*newp)

Given an item in the list, replace it with a different value.

#### Parameters:

*x* The linklist instance.

*p* An item in the list.

*newp* The new value.

#### Returns:

Always returns NULL.

**26.11.2.37 void linklist\_swap (t\_linklist \* *x*, long *a*, long *b*)**

Swap the position of two items in the linked-list, specified by index.

**Parameters:**

- x* The linklist instance.
- a* The index of the first item to swap.
- b* The index of the second item to swap.

## 26.12 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.

#### 26.12.1 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.

#### 26.12.2 Function Documentation

##### 26.12.2.1 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:**

- x* The quickmap instance.
- p1* The (first) key.
- p2* The value (or the second key).

**Returns:**

A Max error code.

**26.12.2.2 void quickmap\_drop (t\_quickmap \* *x*, void \* *p1*, void \* *p2*)**

Drop a pair of keys mapped to each other in the quickmap.

**Parameters:**

- x* The quickmap instance.
- p1* The first key.
- p2* The second key.

**Returns:**

A Max error code.

**26.12.2.3 long quickmap\_lookup\_key1 (t\_quickmap \* *x*, void \* *p1*, void \*\* *p2*)**

Given a (first) key, lookup the value (the second key).

**Parameters:**

- x* The quickmap instance.
- p1* The (first) key.
- p2* The address of a pointer which will hold the resulting key upon return.

**Returns:**

A Max error code.

**26.12.2.4 long quickmap\_lookup\_key2 (t\_quickmap \* *x*, void \* *p1*, void \*\* *p2*)**

Given a (second) key, lookup the value (the first key).

**Parameters:**

- x* The quickmap instance.
- p1* The (second) key.
- p2* The address of a pointer which will hold the resulting key upon return.

**Returns:**

A Max error code.

**26.12.2.5 BEGIN\_USING\_C\_LINKAGE void\* quickmap\_new (void)**

Create a new quickmap object.

**Returns:**

Pointer to the new quickmap object.

**26.12.2.6 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:**

*x* The quickmap instance.

*way* Set to true to make the quickmap readonly (disable thread protection) or false (the default) to enable thread protection.

## 26.13 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)

*Create a new string object.*

#### 26.13.1 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](#)

#### 26.13.2 Function Documentation

##### 26.13.2.1 const char\* [string\\_getptr](#) ([t\\_string](#) \* x)

Create a new string object.

**Parameters:**

*x* The string object instance.

**Returns:**

A pointer to the internally maintained C-string.

**26.13.2.2 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.



## 26.14 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

- 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.*

#### 26.14.1 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

#### 26.14.2 Function Documentation

##### 26.14.2.1 long symobject\_linklist\_match (void \* a, void \* b)

Utility for searching a linklist containing symobjects.

#### Parameters:

- a* (opaque)
- b* (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;

textsym = 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
}
```

#### 26.14.2.2 void\* symobject\_new (t\_symbol \* sym)

The symobject data structure.

##### Parameters:

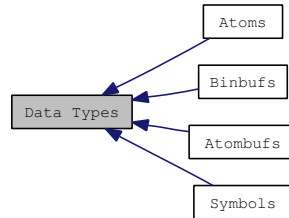
*sym* A symbol with which to initialize the new symobject.

##### Returns:

Pointer to the new symobject instance.

## 26.15 Data Types

Collaboration diagram for Data Types:



### 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.*

### 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.*

### Typedefs

- typedef void \*(\* [method](#) )(void \*,...)  
*Function pointer type for generic methods.*
- typedef long(\* [longmethod](#) )(void \*,...)  
*Function pointer type for methods returning a long.*
- typedef void \*(\* [voidstarvoid](#) )()  
*Function pointer type for a function with no arguments, returning a generic pointer.*

- typedef char \* [t\\_ptr](#)  
*Generic pointer type.*
- typedef char \*\* [t\\_handle](#)  
*Generic pointer-to-a-pointer type.*
- typedef void \* [t\\_vptr](#)  
*Void pointer type.*
- 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.*
- typedef unsigned long [ulong](#)  
*An unsigned long integer.*
- typedef unsigned int [uint](#)  
*An unsigned integer.*
- typedef unsigned short [ushort](#)  
*An unsigned short integer.*
- typedef unsigned char [uchar](#)  
*An unsigned char.*
- typedef long [t\\_max\\_err](#)  
*A Max error code.*

## 26.15.1 Typedef Documentation

### 26.15.1.1 typedef long t\_max\_err

A Max error code. Common error codes are defined in [e\\_max\\_errorcodes](#).

Definition at line 115 of file ext\_obex.h.

## 26.16 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.*

### Defines

- `#define` [ATOM\\_MAX\\_STRLEN](#) (32768)  
*Defines the largest possible string size for an atom.*

### Enumerations

- enum [e\\_max\\_atomtypes](#) {  
[A\\_NOthing](#) = 0,  
[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](#) = 0x41,  
[A\\_USURP](#) = 0x42,  
[A\\_DEFER\\_LOW](#) = 0x43,  
[A\\_USURP\\_LOW](#) = 0x44 }

*the list of officially recognized types, including pseudotypes for commas and semicolons.*

- enum `e_max_atom_gettext_flags` {  
`OBEX_UTIL_ATOM_GETTEXT_DEFAULT` = 0x00000000,  
`OBEX_UTIL_ATOM_GETTEXT_TRUNCATE_ZEROS` = 0x00000001,  
`OBEX_UTIL_ATOM_GETTEXT_SYM_NO_QUOTE` = 0x00000002,  
`OBEX_UTIL_ATOM_GETTEXT_SYM_FORCE_QUOTE` = 0x00000004,  
`OBEX_UTIL_ATOM_GETTEXT_COMMA_DELIM` = 0x00000008,  
`OBEX_UTIL_ATOM_GETTEXT_FORCE_ZEROS` = 0x00000010,  
`OBEX_UTIL_ATOM_GETTEXT_NUM_HI_RES` = 0x00000020 }

*Flags that determine how functions convert atoms into text (C-strings).*

## Functions

- `t_max_err atom_setlong (t_atom *a, 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`.*
- `long atom_getlong (t_atom *a)`  
*Retrieves a long integer value from a `t_atom`.*
- `float atom_getfloat (t_atom *a)`  
*Retrieves a floating point value from a `t_atom`.*
- `t_symbol * atom_getsym (t_atom *a)`  
*Retrieves a `t_symbol *` value from a `t_atom`.*
- `void * atom_getobj (t_atom *a)`  
*Retrieves a generic pointer value from a `t_atom`.*
- `long atom_getcharfix (t_atom *a)`  
*Retrieves an unsigned integer value between 0 and 255 from a `t_atom`.*
- `long atom_gettype (t_atom *a)`  
*Retrieves type from a `t_atom`.*
- `long atom_arg_getlong (long *c, long idx, long ac, 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, 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, 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, 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, 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)`  
*Assign an array of `t_object`\* values to an array of atoms.*
- `t_max_err atom_setparse (long *ac, t_atom **av, char *parsestr)`  
*Parse a C-string into an array of atoms.*
- `t_max_err atom_setformat (long *ac, t_atom **av, char *fmt,...)`  
*Create an array of atoms populated with values using `sprintf`-like syntax.*
- `t_max_err atom_getformat (long ac, t_atom *av, char *fmt,...)`  
*Retrieve values from an array of atoms using `scanf`-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_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, 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` (`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` (short argc, `t_atom` \*argv1, `t_atom` \*argv2)  
*Copy an array of atoms.*
- void `postargs` (short 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.*

## 26.16.1 Enumeration Type Documentation

### 26.16.1.1 enum `e_max_atom_gettext_flags`

Flags that determine how functions convert atoms into text (C-strings).

Enumerator:

`OBEX_UTIL_ATOM_GETTEXT_DEFAULT` default translation rules for getting text from atoms



**OBEX\_UTIL\_ATOM\_GETTEXT\_TRUNCATE\_ZEROS** eliminate redundant zeros for floating point numbers (default used)

**OBEX\_UTIL\_ATOM\_GETTEXT\_SYM\_NO\_QUOTE** don't introduce quotes around symbols with spaces

**OBEX\_UTIL\_ATOM\_GETTEXT\_SYM\_FORCE\_QUOTE** always introduce quotes around symbols (useful for JSON)

**OBEX\_UTIL\_ATOM\_GETTEXT\_COMMA\_DELIM** separate atoms with commas (useful for JSON)

**OBEX\_UTIL\_ATOM\_GETTEXT\_FORCE\_ZEROS** always print the zeros

**OBEX\_UTIL\_ATOM\_GETTEXT\_NUM\_HI\_RES** print more decimal places

Definition at line 1677 of file ext\_obex\_util.h.

### 26.16.1.2 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\_NOTHING** no type, thus no atom

**A\_LONG** long integer

**A\_FLOAT** 32-bit float

**A\_SYM** [t\\_symbol](#) pointer

**A\_OBJ** [t\\_object](#) pointer (for argtype lists; passes the value of sym)

**A\_DEFLONG** long but defaults to zero

**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 deferred 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.

Definition at line 202 of file ext\_mess.h.

## 26.16.2 Function Documentation

### 26.16.2.1 `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.

### 26.16.2.2 `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:

*minsize* The minimum number of atoms that this array will need to contain. This determines the amount of memory allocated.

*ac* The address of a variable that will contain the number of atoms allocated.

*av* The address of a pointer that will be set with the new allocated memory for the atoms.

*alloc* Address of a variable that will be set true if memory is allocated, otherwise false.

#### Returns:

A Max error code.

### 26.16.2.3 `long atom_arg_getdouble (double * c, long idx, long ac, 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:

*c* Pointer to a double variable to receive the atom's data if the function is successful. Otherwise the value is left unchanged.

*idx* 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, *idx* should be set to 2.

*ac* Count of *av*.

*av* Pointer to the first [t\\_atom](#) 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.

**26.16.2.4 long atom\_arg\_getfloat (float \* *c*, long *idx*, long *ac*, t\_atom \* *av*)**

Retrieves the floating point value of a particular [t\\_atom](#) from an atom list, if the atom exists.

**Parameters:**

*c* Pointer to a float variable to receive the atom's data if the function is successful. Otherwise, the value is left unchanged.

*idx* 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, *idx* should be set to 2.

*ac* Count of *av*.

*av* Pointer to the first [t\\_atom](#) 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.

**26.16.2.5 long atom\_arg\_getlong (long \* *c*, long *idx*, long *ac*, t\_atom \* *av*)**

Retrieves the integer value of a particular [t\\_atom](#) from an atom list, if the atom exists.

**Parameters:**

*c* Pointer to a long variable to receive the atom's data if the function is successful.

*idx* 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, *idx* should be set to 2.

*ac* Count of *av*.

*av* Pointer to the first [t\\_atom](#) 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)
{
    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 {
        ...
    }
}

```

#### 26.16.2.6 `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:

- x* The address of a pointer to the object contained in av if it is of the specified class upon return. Otherwise NULL upon return.
- idx* The index of the atom in the array from which to get the object pointer.
- argc* The count of atoms in argv.
- argv* The address to the first of an array of atoms.
- cls* A symbol containing the class name of which the object should be an instance.

##### Returns:

A Max error code.

#### 26.16.2.7 `long atom_arg_getsym (t_symbol **c, long idx, long ac, t_atom *av)`

Retrieves the `t_symbol *` value of a particular `t_atom` from an atom list, if the atom exists.

##### Parameters:

- c* Pointer to a `t_symbol *` variable to receive the atom's data if the function is successful. Otherwise, the value is left unchanged.
- idx* 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, *idx* should be set to 2.
- ac* Count of av.
- av* Pointer to the first `t_atom` 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
}

```

#### 26.16.2.8 **BEGIN\_USING\_C\_LINKAGE** void atom\_copy (short *argc1*, t\_atom \* *argv1*, t\_atom \* *argv2*)

Copy an array of atoms.

##### Parameters:

*argc1* The count of atoms in argv1.

*argv1* The address to the first of an array of atoms that is the source for the copy.

*argv2* 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 [sysmem\\_newptr\(\)](#) or [atom\\_alloc\(\)](#).

#### 26.16.2.9 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:

*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.

#### 26.16.2.10 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:

*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.

**26.16.2.11 long atom\_getcharfix (t\_atom \* a)**

Retrieves an unsigned integer value between 0 and 255 from a [t\\_atom](#).

**Parameters:**

*a* Pointer to a [t\\_atom](#) 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.

**26.16.2.12 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:**

*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.

**26.16.2.13 float atom\_getfloat (t\_atom \* a)**

Retrieves a floating point value from a [t\\_atom](#).

**Parameters:**

*a* Pointer to a [t\\_atom](#) 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.

**26.16.2.14 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:**

- 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.

**26.16.2.15 t\_max\_err atom\_getformat (long ac, t\_atom \* av, char \* fmt, ...)**

Retrieve values from an array of atoms using sscanf-like syntax. [atom\\_getformat\(\)](#) supports `clfdsoaCLFD-SOA` 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:**

- ac* The number of atoms to parse in *av*.
- av* The address of the first [t\\_atom](#) pointer in an array to parse.
- fmt* 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 *fmt* string.

**Returns:**

A Max error code.

**See also:**

[atom\\_setformat\(\)](#)

**26.16.2.16 long atom\_getlong (t\_atom \* a)**

Retrieves a long integer value from a [t\\_atom](#).

**Parameters:**

*a* Pointer to a [t\\_atom](#) 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.

**26.16.2.17 t\_max\_err atom\_getlong\_array (long ac, t\_atom \* av, long count, 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.

**26.16.2.18 void\* atom\_getobj (t\_atom \* a)**

Retrieves a generic pointer value from a [t\\_atom](#).

**Parameters:**

*a* Pointer to a [t\\_atom](#) 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.

**26.16.2.19 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:**

- 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.

**26.16.2.20 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:**

- av* A pointer to the atom from which to get the [t\\_object](#).
- cls* 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.

**26.16.2.21 t\_symbol\* atom\_getsym (t\_atom \* a)**

Retrieves a [t\\_symbol](#) \* value from a [t\\_atom](#).

**Parameters:**

- a* Pointer to a [t\\_atom](#) 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.

**26.16.2.22 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:**

- 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.

**26.16.2.23 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:**

*ac* The number of atoms to fetch in *av*.

*av* The address of the first [t\\_atom](#) pointer in an array to retrieve.

*textsize* The size of the string to which the atoms will be formatted and copied.

*text* The address of the string to which the text will be written.

*flags* Determines the rules by which atoms will be translated into text. Values are bit mask as defined by [e\\_max\\_atom\\_gettext\\_flags](#).

**Returns:**

A Max error code.

**See also:**

[atom\\_setparse\(\)](#)

**26.16.2.24 long atom\_gettype (t\_atom \* *a*)**

Retrieves type from a [t\\_atom](#).

**Parameters:**

*a* Pointer to a [t\\_atom](#) whose type is of interest

**Returns:**

This function returns the type of the specified [t\\_atom](#) as defined in [e\\_max\\_atomtypes](#)

**26.16.2.25 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:**

*ac* 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 *av*.

*av* The address to the first of an array of allocated atoms.

*count* The number of values in the array specified by *vals*.

*vals* The array from which to copy the values into the array of atoms at *av*.

**Returns:**

A Max error code.

**26.16.2.26 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:**

- ac* 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*.
- av* The address to the first of an array of allocated atoms.
- count* The number of values in the array specified by *vals*.
- vals* The array from which to copy the values into the array of atoms at *av*.

**Returns:**

A Max error code.

**26.16.2.27 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:**

- ac* 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*.
- av* The address to the first of an array of allocated atoms.
- count* The number of values in the array specified by *vals*.
- vals* The array from which to copy the values into the array of atoms at *av*.

**Returns:**

A Max error code.

**26.16.2.28 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:**

- a* Pointer to a [t\\_atom](#) whose value and type will be modified
- b* Floating point value to copy into the [t\\_atom](#)

**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.

**26.16.2.29 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:**

- ac* The number of atoms to try to fetch from the array of floats. You should have at least this number of atoms allocated in *av*.
- av* The address to the first of an array of allocated atoms.
- count* The number of values in the array specified by *vals*.
- vals* The array from which to copy the values into the array of atoms at *av*.

**Returns:**

A Max error code.

**26.16.2.30 t\_max\_err atom\_setformat (long \* *ac*, t\_atom \*\* *av*, char \* *fmt*, ...)**

Create an array of atoms populated with values using sprintf-like syntax. [atom\\_setformat\(\)](#) supports cldf-soaCLFDSOA 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:**

- ac* The address of a variable to hold the number of returned atoms.
- av* The address of a [t\\_atom](#) pointer to which memory may be allocated and atoms copied.
- fmt* 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\(\)](#)

**26.16.2.31 t\_max\_err atom\_setlong (t\_atom \* *a*, long *b*)**

Inserts an integer into a [t\\_atom](#) and change the *t\_atom*'s type to [A\\_LONG](#).

**Parameters:**

- a* Pointer to a [t\\_atom](#) whose value and type will be modified
- b* Integer value to copy into the [t\\_atom](#)

**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 [db\\_open\(\)](#).

**26.16.2.32 t\_max\_err atom\_setlong\_array (long *ac*, t\_atom \* *av*, long *count*, long \* *vals*)**

Assign an array of long integer values to an array of atoms.

**Parameters:**

*ac* The number of atoms to try to fetch from the array of longs. You should have at least this number of atoms allocated in *av*.

*av* The address to the first of an array of allocated atoms.

*count* The number of values in the array specified by *vals*.

*vals* The array from which to copy the values into the array of atoms at *av*.

**Returns:**

A Max error code.

**26.16.2.33 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:**

*a* Pointer to a [t\\_atom](#) whose value and type will be modified

*b* Pointer value to copy into the [t\\_atom](#)

**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.

**26.16.2.34 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:**

*ac* The number of atoms to try to fetch from the array of objects. You should have at least this number of atoms allocated in *av*.

*av* The address to the first of an array of allocated atoms.

*count* The number of values in the array specified by *vals*.

*vals* The array from which to copy the values into the array of atoms at *av*.

**Returns:**

A Max error code.

### 26.16.2.35 `t_max_err atom_setparse (long * ac, t_atom ** av, 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:

- ac* The address of a variable to hold the number of returned atoms.
- av* The address of a [t\\_atom](#) pointer to which memory may be allocated and atoms copied.
- parsestr* 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");
```

### 26.16.2.36 `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:

- a* Pointer to a [t\\_atom](#) whose value and type will be modified
- b* Pointer to a [t\\_symbol](#) to copy into the [t\\_atom](#)

#### 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 [db\\_open\(\)](#).

### 26.16.2.37 `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:

- ac* The number of atoms to try to fetch from the array of symbols. You should have at least this number of atoms allocated in *av*.
- av* The address to the first of an array of allocated atoms.
- count* The number of values in the array specified by *vals*.
- vals* The array from which to copy the values into the array of atoms at *av*.

#### Returns:

A Max error code.

**26.16.2.38 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.

**26.16.2.39 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.

**26.16.2.40 long atomisstring (t\_atom \* a)**

Determines whether or not an atom represents a [t\\_string](#) object.

**Parameters:**

*a* The address of the atom to test.

**Returns:**

Returns true if the [t\\_atom](#) contains a valid [t\\_string](#) object.

**26.16.2.41 void postargs (short argc, t\_atom \* argv)**

Print the contents of an array of atoms to the Max window.

**Parameters:**

*argc* The count of atoms in argv.

*argv* The address to the first of an array of atoms.

## 26.17 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

- 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](#).*

#### 26.17.1 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).

#### 26.17.2 Function Documentation

##### 26.17.2.1 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.

##### 26.17.2.2 void\* atombuf\_new (long argc, t\_atom \* argv)

Use [atombuf\\_new\(\)](#) to create a new Atombuf from an array of [t\\_atoms](#).



**Parameters:**

*argc* Number of `t_atoms` in the `argv` array. May be 0.

*argv* 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.

**26.17.2.3 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:**

*x* Pointer to existing atombuf variable. The variable will be replaced by a new Atombuf containing the converted text.

*text* Handle to the text to be converted. It need not be zero-terminated.

*size* Number of characters in the text.

## 26.18 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 (void *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 (void *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 (void *x, long *p1, long *p2, t_atom *ap)`  
*Use `binbuf_getatom` to retrieve a single Atom from a Binbuf.*
- `short binbuf_text (void *x, char **srcText, long n)`  
*Use `binbuf_text()` to convert a text handle to a Binbuf.*
- `short binbuf_totext (void *x, char **dstText, long *sizep)`  
*Use `binbuf_totext()` to convert a Binbuf into a text handle.*
- `void binbuf_set (void *x, t_symbol *s, short argc, t_atom *argv)`  
*Use `binbuf_set()` to change the entire contents of a Binbuf.*
- `void binbuf_append (void *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 Atom from a text buffer.*

### 26.18.1 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.

## 26.18.2 Function Documentation

### 26.18.2.1 void binbuf\_append (void \* *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.

### 26.18.2.2 void\* binbuf\_eval (void \* *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.

### 26.18.2.3 short binbuf\_getatom (void \* *x*, long \* *p1*, long \* *p2*, t\_atom \* *ap*)

Use binbuf\_getatom to retrieve a single Atom from a Binbuf.

**Parameters:**

- x* Binbuf containing the desired [t\\_atom](#).
- p1* Offset into the Binbuf's array of types. Modified to point to the next [t\\_atom](#).
- p2* Offset into the Binbuf's array of data. Modified to point to the next [t\\_atom](#).
- ap* Location of a [t\\_atom](#) 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
}
```

#### 26.18.2.4 void binbuf\_insert (void \* *x*, t\_symbol \* *s*, short *argc*, t\_atom \* *argv*)

Use [binbuf\\_insert\(\)](#) to append a Max message to a Binbuf adding a semicolon.

##### Parameters:

- x* Binbuf to receive the items.
- s* Ignored. Pass NULL.
- argc* Count of items in the argv array.
- argv* 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.

#### 26.18.2.5 void\* binbuf\_new (void)

Use [binbuf\\_new\(\)](#) to create and initialize a Binbuf.

##### Returns:

Returns a new binbuf if successful, otherwise NULL.

#### 26.18.2.6 void binbuf\_set (void \* *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:

- x* Binbuf to receive the items.
- s* Ignored. Pass NULL.
- argc* Count of items in the argv array.
- argv* Array of t\_atoms to put in the Binbuf.

#### 26.18.2.7 short binbuf\_text (void \* *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:

- x* Binbuf to contain the converted text. It must have already been created with binbuf\_new. Its previous contents are destroyed.

*srcText* Handle to the text to be converted. It need not be terminated with a 0.

*n* 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.

#### 26.18.2.8 short binbuf\_totext (void \* *x*, char \*\* *dstText*, long \* *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 [binbuf\\_totext\(\)](#) 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.

#### 26.18.2.9 void binbuf\_vinsert (void \* *x*, char \* *fmt*, ...)

Use [binbuf\\_vinsert\(\)](#) to append a Max message to a Binbuf adding a semicolon.

#### Parameters:

*x* Binbuf containing the desired Atom.

**fmt** A C-string containing one or more letters corresponding to the types of each element of the message. s for [t\\_symbol\\*](#), 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, "sll", 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.

### 26.18.2.10 short readatom (char \* outstr, char \*\* text, long \* n, long e, t\_atom \* ap)

Use [readatom\(\)](#) to read a single Atom from a text buffer.

#### Parameters:

**outstr** C-string of 256 characters that will receive the next text item read from the buffer.

**text** Handle to the text buffer to be read.

**n** Starts at 0, and is modified by readatom to point to the next item in the text buffer.

**e** Number of characters in text.

**ap** Where the resulting Atom 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 Atom
}
```

An alternative to using `readatom` is to turn your text into a Binbuf using `binbuf_text()`, then call `binbuf_getatom()` in a loop.

## 26.19 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](#) (char \*s)

*Given a C-string, fetch the matching [t\\_symbol](#) pointer from the symbol table, generating the symbol if necessary.*

#### 26.19.1 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;
```



## 26.19.2 Function Documentation

### 26.19.2.1 `t_symbol*` gensym (`char * s`)

Given a C-string, fetch the matching `t_symbol` pointer from the symbol table, generating the symbol if neccessary.

**Parameters:**

*s* 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 `db_open()`, `db_view_create()`, `db_view_getresult()`, `db_view_remove()`, and `db_view_setquery()`.

## 26.20 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.*

### Defines

- #define [MAX\\_PATH\\_CHARS](#) 2048  
*The size you should use when allocating strings for full paths.*
- #define [MAX\\_FILENAME\\_CHARS](#) 512  
*The size you should use when allocating strings for filenames.*

### Typedefs

- typedef void \* [t\\_filehandle](#)  
*A `t_filehandle` is a cross-platform way of referring to an open file.*

### Enumerations

- enum [e\\_max\\_path\\_styles](#) {  
    [PATH\\_STYLE\\_MAX](#) = 0,  
    [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](#) = 0,  
    [PATH\\_TYPE\\_ABSOLUTE](#),  
    [PATH\\_TYPE\\_RELATIVE](#),

```
PATH_TYPE_BOOT,
PATH_TYPE_C74,
PATH_TYPE_PATH }
```

*Constants that determine the output of `path_nameconform()`.*

- enum `e_max_fileinfo_flags` {  
`PATH_FILEINFO_ALIAS` = 1,  
`PATH_FILEINFO_FOLDER` = 2,  
`PATH_FILEINFO_PACKAGE` = 4 }

*Flags used to represent properties of a file in a `t_fileinfo` struct.*

- enum `e_max_path_folder_flags` {  
`PATH_REPORTPACKAGEASFOLDER` = 1,  
`PATH_FOLDER_SNIFF` = 2 }

*Flags used by functions such as `path_foldernextfile()` and `path_openfolder()`.*

- enum `e_max_openfile_permissions` {  
`PATH_READ_PERM` = 1,  
`PATH_WRITE_PERM` = 2,  
`PATH_RW_PERM` = 3 }

*Permissions or mode with which to open a file.*

- enum `e_max_sysfile_posmodes` {  
`SYSFILE_ATMARK` = 0,  
`SYSFILE_FROMSTART`,  
`SYSFILE_FROMLEOF`,  
`SYSFILE_FROMMARK` }

*Modes used by `sysfile_setpos()`.*

- enum `e_max_sysfile_textflags` {  
`TEXT_LB_NATIVE` = 0x00000001L,  
`TEXT_LB_MAC` = 0x00000002L,  
`TEXT_LB_PC` = 0x00000004L,  
`TEXT_LB_UNIX` = 0x00000008L,  
`TEXT_ENCODING_USE_FILE` = 0x00000100L,  
`TEXT_NULL_TERMINATE` = 0x00000200L }

*Flags used reading and writing text files.*

## Functions

- short `path_getapppath` (void)  
*Retrieve the Path ID of the Max application.*
- short `locatefile` (char \*name, short \*outvol, short \*binflag)

*Find a Max document by name in the search path.*

- short [locatefiletype](#) (char \*name, short \*outvol, long filetype, long creator)  
*Find a Max document by name in the search path.*
- short [locatefile\\_extended](#) (char \*name, short \*outvol, long \*outtype, long \*filetypelist, short numtypes)  
*Find a Max document by name in the search path.*
- short [path\\_resolvefile](#) (char \*name, 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](#) (char \*name, short path, void \*info)  
*Retrieve a [t\\_fileinfo](#) structure from a file/path combination.*
- short [path\\_topathname](#) (short path, char \*file, char \*name)  
*Create a fully qualified file name from a Path ID/file name combination.*
- short [path\\_frompathname](#) (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, unsigned long \*date)  
*Determine the modification date of the selected path.*
- short [path\\_getfilemoddate](#) (char \*filename, short path, unsigned long \*date)  
*Determine the modification date of the selected file.*
- void \* [path\\_openfolder](#) (short path)  
*Prepare a directory for iteration.*
- short [path\\_foldernextfile](#) (void \*\*xx, long \*filetype, char \*name, short descend)  
*Get the next file in the directory.*
- void [path\\_closefolder](#) (void \*x)  
*Complete a directory iteration.*
- short [path\\_opensysfile](#) (char \*name, short path, [t\\_filehandle](#) \*ref, short perm)  
*Open a file given a filename and Path ID.*
- short [path\\_createsysfile](#) (char \*name, short path, long type, [t\\_filehandle](#) \*ref)  
*Create a file given a type code, a filename, and a Path ID.*
- short [path\\_nameconform](#) (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](#) (short path, 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.*
- short [open\\_dialog](#) (char \*name, short \*volptr, long \*typeptr, long \*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, long \*type, long \*typelist, short numtypes)  
*Present the user with the standard save file dialog with your own list of file types.*
- void [open\\_promptset](#) (char \*s)  
*Use [open\\_promptset\(\)](#) to add a prompt message to the open file dialog displayed by [open\\_dialog\(\)](#).*
- void [saveas\\_promptset](#) (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, short path, char \*filename)  
*Create a new filewatcher.*
- void [fileusage\\_addfile](#) (void \*w, long flags, char \*name, short path)  
*Add a file to a collective.*
- long [sysfile\\_close](#) ([t\\_filehandle](#) f)  
*Close a file opened with [sysfile\\_open\(\)](#).*
- long [sysfile\\_read](#) ([t\\_filehandle](#) f, long \*count, void \*bufptr)  
*Read a file from disk.*
- long [sysfile\\_readtohandle](#) ([t\\_filehandle](#) f, char \*\*\*h)  
*Read the contents of a file into a handle.*
- long [sysfile\\_readtoptr](#) ([t\\_filehandle](#) f, char \*\*p)  
*Read the contents of a file into a pointer.*
- long [sysfile\\_write](#) ([t\\_filehandle](#) f, long \*count, const void \*bufptr)  
*Write part of a file to disk.*
- long [sysfile\\_seteof](#) ([t\\_filehandle](#) f, long logeof)  
*Set the size of a file handle.*
- long [sysfile\\_geteof](#) ([t\\_filehandle](#) f, long \*logeof)  
*Get the size of a file handle.*
- long [sysfile\\_setpos](#) ([t\\_filehandle](#) f, long mode, long offset)

*Set the current file position of a file handle.*

- long [sysfile\\_getpos](#) ([t\\_filehandle](#) f, long \*filepos)

*Get the current file position of a file handle.*

- long [sysfile\\_spoolcopy](#) ([t\\_filehandle](#) src, [t\\_filehandle](#) dst, long size)

*Copy the contents of one file handle to another file handle.*

- long [sysfile\\_readtextfile](#) ([t\\_filehandle](#) f, [t\\_handle](#) htext, long maxlen, long flags)

*Read a text file from disk.*

- long [sysfile\\_writetextfile](#) ([t\\_filehandle](#) f, [t\\_handle](#) htext, long flags)

*Write a text file to disk.*

- short [sysfile\\_openhandle](#) (char \*\*h, long flags, [t\\_filehandle](#) \*fh)

*Create a [t\\_filehandle](#) from a pre-existing handle.*

- short [sysfile\\_openptrsize](#) (char \*p, long length, long flags, [t\\_filehandle](#) \*fh)

*Create a [t\\_filehandle](#) from a pre-existing pointer.*

### 26.20.1 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.

## 26.20.2 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.

## 26.20.3 Example: filebyte (notes from the IRCAM workshop)

### 26.20.3.1 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

### 26.20.3.2 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

### 26.20.3.3 File Names

- C string
- Max 5 filenames are UTF-8
- Max 5 supports long (unicode) filenames on both Mac and Windows

### 26.20.3.4 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](#)

## 26.20.4 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.

## 26.20.5 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.

## 26.20.6 Define Documentation

### 26.20.6.1 #define MAX\_FILENAME\_CHARS 512

The size you should use when allocating strings for filenames. At the time of this writing it supports up to 256 UTF chars

Definition at line 28 of file [ext\\_path.h](#).

## 26.20.7 Typedef Documentation

### 26.20.7.1 typedef void\* 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](#).

Definition at line 15 of file [ext\\_sysfile.h](#).

## 26.20.8 Enumeration Type Documentation

### 26.20.8.1 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)
```

Definition at line 88 of file [ext\\_path.h](#).



### 26.20.8.2 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.

Definition at line 111 of file ext\_path.h.

### 26.20.8.3 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

Definition at line 101 of file ext\_path.h.

### 26.20.8.4 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

Definition at line 45 of file ext\_path.h.

### 26.20.8.5 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  
*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

Definition at line 67 of file ext\_path.h.

**26.20.8.6 enum e\_max\_sysfile\_posmodes**

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.

Definition at line 20 of file ext\_sysfile.h.

**26.20.8.7 enum e\_max\_sysfile\_textflags**

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 [sysfile\\_readtextfile\(\)](#) with a NULL character.

Definition at line 30 of file ext\_sysfile.h.

**26.20.9 Function Documentation****26.20.9.1 void fileusage\_addfile (void \* *w*, long *flags*, char \* *name*, short *path*)**

Add a file to a collective.

**Parameters:**

*w* Handle for the collective builder.  
*flags* If flags == 1, copy this file to support folder of an app instead of to the collective in an app.  
*name* The name of the file.  
*path* The path of the file to add.

### 26.20.9.2 void\* filewatcher\_new (t\_object \* owner, short path, 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:

**owner** Your object. This object will receive the message "filechanged" when the watcher sees a change in the file or folder.

**path** The path in which the file being watched resides, or the path of the folder being watched.

**filename** 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);
```

### 26.20.9.3 short locatefile (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 Patchex 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:

**name** A C string that is the name of the file to look for.

**outvol** The Path ID containing the location of the file if it is found.

**binflag** 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\(\)](#)

#### 26.20.9.4 short locatefile\_extended (char \* *name*, short \* *outvol*, long \* *outtype*, long \* *filetypelist*, short *numtypes*)

Find a Max document by name in the search path. This is the preferred method for file searching since its introduction in Max version 4.

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

##### Version:

4.0

##### Parameters:

*name* The file name for the search, receives actual filename.

*outvol* The Path ID of the file (if found).

*outtype* The file type of the file (if found).

*filetypelist* The file type(s) that you are searching for.

*numtypes* The number of file types in the typelist array (1 if a single entry).

##### Returns:

If a file is found with the name specified by filename, locatefile returns 0, otherwise it returns non-zero.

##### 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, typelist, 1);
```

#### 26.20.9.5 short locatefiletype (char \* *name*, short \* *outvol*, long *filetype*, long *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:

*name* A C string that is the name of the file to look for.

*outvol* The Path ID containing the location of the file if it is found.

*filetype* The filetype of the file to look for. If you pass 0L, files of all filetypes are considered.

**creator** 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\(\)](#)

#### 26.20.9.6 short open\_dialog (char \* name, short \* volptr, long \* typeptr, long \* 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:**

**name** 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](#).

**volptr** Receives the Path ID of the file the user wants to open.

**typeptr** The file type of the file the user wants to open.

**types** 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.

**ntypes** 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\(\)](#)

#### 26.20.9.7 void open\_promptset (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:**

**s** A C-string containing the prompt you wish to display in the dialog box.

**Returns:**

Ignore.

**See also:**

[open\\_dialog\(\)](#)

### 26.20.9.8 void path\_closefolder (void \* *x*)

Complete a directory iteration.

#### Parameters:

*x* The “folder state” value originally returned by [path\\_openfolder\(\)](#).

### 26.20.9.9 short path\_createsysfile (char \* *name*, short *path*, long *type*, t\_filehandle \* *ref*)

Create a file given a type code, a filename, and a Path ID.

#### Parameters:

*name* The name of the file to be opened.

*path* The Path ID of the file to be opened.

*type* The file type of the created file.

*ref* A [t\\_filehandle](#) reference to the opened file will be returned in this parameter.

#### Returns:

An error code.

### 26.20.9.10 short path\_fileinfo (char \* *name*, short *path*, void \* *info*)

Retrieve a [t\\_fileinfo](#) structure from a file/path combination.

#### Parameters:

*name* The file name to be queried.

*path* The Path ID of the file.

*info* The address of a [t\\_fileinfo](#) structure to contain the file information.

#### Returns:

Returns 0 if successful, otherwise it returns an OS-specific error code.

### 26.20.9.11 short path\_foldernextfile (void \* *xx*, long \* *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:

*xx* The “folder state” value returned by [path\\_openfolder\(\)](#).

*filetype* Contains the file type of the file type on return.

*name* Contains the file name of the next file on return.

*descend* Unused.

**Returns:**

Returns non-zero if successful, and zero when there are no more files.

**See also:**

[e\\_max\\_path\\_folder\\_flags](#)

**26.20.9.12 short path\_frompathname (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:**

*name* The extended file path to be converted.

*path* Contains the Path ID on return.

*filename* Contains the file name on return.

**Returns:**

Returns 0 if successful, otherwise it returns an OS-specific error code.

**26.20.9.13 short path\_getapppath (void)**

Retrieve the Path ID of the Max application.

**Returns:**

The path id.

**26.20.9.14 short path\_getdefault (void)**

Retrieve the Path ID of the default search path.

**Returns:**

The path id of the default search path.

**26.20.9.15 short path\_getfilemoddate (char \* *filename*, short *path*, unsigned long \* *date*)**

Determine the modification date of the selected file.

**Parameters:**

*filename* The name of the file to query.

*path* The Path ID of the file.

*date* The last modification date of the file upon return.

**Returns:**

An error code.

**26.20.9.16 short path\_getmoddate (short *path*, unsigned long \* *date*)**

Determine the modification date of the selected path.

**Parameters:**

*path* The Path ID of the directory to check.

*date* The last modification date of the directory.

**Returns:**

An error code.

**26.20.9.17 short path\_nameconform (char \* *src*, char \* *dst*, long *style*, long *type*)**

Convert a source path string to destination path string using the specified style and type.

**Parameters:**

*src* A pointer to source character string to be converted.

*dst* A pointer to destination character string.

*style* The destination filepath style, as defined in [e\\_max\\_path\\_styles](#)

*type* The destination filepath type, as defined in [e\\_max\\_path\\_types](#)

**Returns:**

An error code.

**See also:**

[MAX\\_PATH\\_CHARS](#)

Referenced by [db\\_open\(\)](#).

**26.20.9.18 void\* path\_openfolder (short *path*)**

Prepare a directory for iteration.

**Parameters:**

*path* 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.



**26.20.9.19 short path\_opensysfile (char \* *name*, short *path*, t\_filehandle \* *ref*, short *perm*)**

Open a file given a filename and Path ID.

**Parameters:**

*name* The name of the file to be opened.

*path* The Path ID of the file to be opened.

*ref* A [t\\_filehandle](#) reference to the opened file will be returned in this parameter.

*perm* The permission for the opened file as defined in [e\\_max\\_openfile\\_permissions](#).

**Returns:**

An error code.

**26.20.9.20 short path\_resolvefile (char \* *name*, 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:**

*name* A file name (which may be fully or partially qualified), will contain the file name on return.

*path* The Path ID to be resolved.

*outpath* The Path ID of the returned file name.

**Returns:**

Returns 0 if successful.

**26.20.9.21 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:**

*path* 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).

*recursive* 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.

**26.20.9.22 short path\_topathname (short *path*, 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:**

*path* The path to be used.  
*file* The file name to be used.  
*name* Loaded with the fully extended file name on return.

**Returns:**

Returns 0 if successful, otherwise it returns an OS-specific error code.

**26.20.9.23 short\_path\_topotentialname (short path, 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:**

*path* The path to be used.  
*file* The file name to be used.  
*name* Loaded with the fully extended file name on return.  
*check* 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\(\)](#)

**26.20.9.24 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:**

*filename* 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](#).  
*path* If the user decides to save the file, the Path ID of the location chosen is returned here.  
*binptr* 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\(\)](#)

**26.20.9.25 void saveas\_promptset (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:**

*s* A C-string containing the prompt you wish to display in the dialog box.

**Returns:**

Ignore.

**See also:**

[open\\_dialog\(\)](#)

**26.20.9.26 short saveasdialog\_extended (char \* name, short \* vol, long \* type, long \* 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:**

**name** 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`.

**vol** If the user decides to save the file, the Path ID of the location chosen is returned here.

**type** Returns the type of file chosen by the user.

**typelist** The list of types provided to the user.

**numtypes** 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\(\)](#)

**26.20.9.27 long 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:**

*f* The `t_filehandle` structure of the file the user wants to close.

**Returns:**

An error code.

**26.20.9.28 long sysfile\_geteof (t\_filehandle f, long \* 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:**

*f* The file's `t_filehandle` structure.

*logeof* The file size in bytes is returned to this value.

**Returns:**

An error code.

**26.20.9.29 long sysfile\_getpos (t\_filehandle *f*, long \**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:**

*f* The file's [t\\_filehandle](#) structure.

*filepos* The address of a variable to hold the current file position of file handle in bytes.

**Returns:**

An error code.

**26.20.9.30 short sysfile\_openhandle (char \*\**h*, long *flags*, t\_filehandle \**fh*)**

Create a [t\\_filehandle](#) from a pre-existing handle.

**Parameters:**

*h* A handle for some data.

*flags* Pass 0 (additional flags are private).

*fh* The address of a [t\\_filehandle](#) which will be allocated.

**Returns:**

An error code.

**26.20.9.31 short sysfile\_openptrsize (char \**p*, long *length*, long *flags*, t\_filehandle \**fh*)**

Create a [t\\_filehandle](#) from a pre-existing pointer.

**Parameters:**

*p* A pointer to some data.

*length* The size of *p*.

*flags* Pass 0 (additional flags are private).

*fh* The address of a [t\\_filehandle](#) which will be allocated.

**Returns:**

An error code.

**26.20.9.32 long sysfile\_read (t\_filehandle *f*, long \**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:**

- f* The [t\\_filehandle](#) structure of the file the user wants to open.
- count* 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.
- bufptr* Pointer to the buffer that the data will be read into.

**Returns:**

An error code.

**26.20.9.33 long sysfile\_readtextfile (t\_filehandle *f*, t\_handle *h*text, long *maxlen*, long *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 *h*text handle, performing linebreak translation if set in *flags*.

**Parameters:**

- f* The [t\\_filehandle](#) structure of the text file the user wants to open.
- h*text Handle that the data will be read into.
- maxlen* The maximum length in bytes to be read into the handle. Passing the value 0L indicates no maximum (i.e. read the entire file).
- flags* Flags to set linebreak translation as defined in [e\\_max\\_sysfile\\_textflags](#).

**Returns:**

An error code.

**26.20.9.34 long sysfile\_readtohandle (t\_filehandle *f*, char \*\*\* *h*)**

Read the contents of a file into a handle.

**Parameters:**

- f* The open [t\\_filehandle](#) structure to read into the handle.
- h* 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 [systemem\\_freehandle\(\)](#).

**26.20.9.35 long sysfile\_readtoptr (t\_filehandle *f*, char \*\* *p*)**

Read the contents of a file into a pointer.

**Parameters:**

- f* The open [t\\_filehandle](#) structure to read into the handle.

*p* 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\(\)](#).

**26.20.9.36 long sysfile\_seteof (t\_filehandle *f*, long *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:**

*f* The file’s [t\\_filehandle](#) structure.

*logeof* The file size in bytes.

**Returns:**

An error code.

**26.20.9.37 long sysfile\_setpos (t\_filehandle *f*, long *mode*, long *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 [e\\_max\\_sysfile\\_posmodes](#).

**Parameters:**

*f* The file’s [t\\_filehandle](#) structure.

*mode* Mode from which the offset will be calculated, as defined in [e\\_max\\_sysfile\\_posmodes](#).

*offset* The offset in bytes relative to the mode.

**Returns:**

An error code.

**26.20.9.38 long sysfile\_spoolcopy (t\_filehandle *src*, t\_filehandle *dst*, long *size*)**

Copy the contents of one file handle to another file handle.

**Parameters:**

*src* The file handle from which to copy.

*dst* The file handle to which the copy is written.

*size* The number of bytes to copy. If 0 the size of *src* will be used.

**Returns:**

An error code.

**26.20.9.39** `long sysfile_write (t_filehandle f, long * 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:**

*f* The `t_filehandle` structure of the file to which the user wants to write.

*count* 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.

*bufptr* Pointer to the buffer that the data will be read from.

**Returns:**

An error code.

**26.20.9.40** `long sysfile_writetextfile (t_filehandle f, t_handle htext, long 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:**

*f* The `t_filehandle` structure of the text file to which the user wants to write.

*htext* Handle that the data that will be read from.

*flags* Flags to set linebreak translation as defined in `e_max_sysfile_textflags`.

**Returns:**

An error code.



## 26.21 Memory Management

In the past, Max has provided two separate APIs for memory management.

### Defines

- #define [MM\\_UNIFIED](#)

*This macro being defined means that [getbytes](#) and [sysmem](#) APIs for memory management are unified.*

### Functions

- char \* [getbytes](#) (short size)  
*Allocate small amounts of non-relocatable memory.*
- void [freebytes](#) (void \*b, short size)  
*Free memory allocated with [getbytes](#)().*
- char \* [getbytes16](#) (short 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, short 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 [disposhandle](#) (char \*\*h)  
*Free the memory used by a handle you no longer need.*
- [t\\_ptr](#) [sysmem\\_newptr](#) (long size)  
*Allocate memory.*
- [t\\_ptr](#) [sysmem\\_newptrclear](#) (long size)  
*Allocate memory and set it to zero.*
- [t\\_ptr](#) [sysmem\\_resizeptr](#) (void \*ptr, long newsize)  
*Resize an existing pointer.*
- [t\\_ptr](#) [sysmem\\_resizeptrclear](#) (void \*ptr, long newsize)  
*Resize an existing pointer and clear it.*
- long [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, long bytes)  
*Copy memory the contents of one pointer to another pointer.*
- `t_handle` `sysmem_newhandle` (long size)  
*Allocate a handle (a pointer to a pointer).*
- `t_handle` `sysmem_newhandleclear` (unsigned long size)  
*Allocate a handle (a pointer to a pointer) whose memory is set to zero.*
- long `sysmem_resizehandle` (`t_handle` handle, long newsize)  
*Resize an existing handle.*
- long `sysmem_handlesize` (`t_handle` handle)  
*Find the size of a handle.*
- void `sysmem_freehandle` (`t_handle` handle)  
*Free memory allocated with `sysmem_newhandle()`.*
- long `sysmem_lockhandle` (`t_handle` handle, long lock)  
*Set the locked/unlocked state of a handle.*
- long `sysmem_ptrandhand` (void \*p, `t_handle` h, long size)  
*Add memory to an existing handle and copy memory to the resized portion from a pointer.*
- long `sysmem_ptrbeforehand` (void \*p, `t_handle` h, unsigned long size)  
*Add memory to an existing handle and copy memory to the resized portion from a pointer.*
- long `sysmem_nullterminatehandle` (`t_handle` h)  
*Add a null terminator to a handle.*

### 26.21.1 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.

## 26.21.2 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).

## 26.21.3 Define Documentation

### 26.21.3.1 #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 old max targets.

Definition at line 30 of file ext.h.

## 26.21.4 Function Documentation

### 26.21.4.1 void disposhandle (char \*\* *h*)

Free the memory used by a handle you no longer need.

#### Parameters:

*h* The handle to dispose.

#### See also:

[sysmem\\_freehandle\(\)](#)

### 26.21.4.2 void freebytes (void \* *b*, short *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:

*b* A pointer to the block of memory previously allocated that you want to free.

*size* The size the block specified (as parameter *b*) in bytes.

### 26.21.4.3 void freebytes16 (char \* *mem*, short *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:

*mem* A pointer to the block of memory previously allocated that you want to free.

*size* The size the block specified (as parameter *b*) 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\(\)](#).

**26.21.4.4 char\* getbytes (short 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:**

*size* The size to allocate in bytes (up to 32767 bytes).

**Returns:**

A pointer to the allocated memory.

**26.21.4.5 char\* getbytes16 (short 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:**

*size* 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\(\)](#).

**26.21.4.6 short growhandle (void \* h, long size)**

Change the size of a handle.

**Parameters:**

*h* The handle to resize.

*size* The new size to allocate in bytes.

**Returns:**

Ignored.

**See also:**

[sysmem\\_resizehandle\(\)](#)

**26.21.4.7 char\*\* newhandle (long size)**

Allocate relocatable memory.

**Parameters:**

*size* The size to allocate in bytes.

**Returns:**

The allocated handle.

**See also:**

[sysmem\\_newhandle\(\)](#)

**26.21.4.8 void sysmem\_copyptr (const void \* src, void \* dst, long 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:**

*src* A pointer to the memory whose bytes will be copied.

*dst* A pointer to the memory where the data will be copied.

*bytes* The size in bytes of the data to be copied.

**26.21.4.9 void sysmem\_freehandle (t\_handle handle)**

Free memory allocated with [sysmem\\_newhandle\(\)](#).

**Parameters:**

*handle* The handle whose memory will be freed.

**26.21.4.10 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:**

*ptr* The pointer whose memory will be freed.

**26.21.4.11 long sysmem\_handlesize (t\_handle handle)**

Find the size of a handle. This function is similar to GetHandleSize().

**Parameters:**

*handle* The handle whose size will be queried.

**Returns:**

The number of bytes allocated to the specified handle.

#### 26.21.4.12 `long sysmem_lockhandle (t_handle handle, long 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:**

*handle* The handle that will be locked.

*lock* The new lock state of the handle.

**Returns:**

The previous lock state.

#### 26.21.4.13 `t_handle sysmem_newhandle (long 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:**

*size* The size of the handle in bytes that will be allocated.

**Returns:**

A new [t\\_handle](#).

#### 26.21.4.14 `t_handle sysmem_newhandleclear (unsigned long size)`

Allocate a handle (a pointer to a pointer) whose memory is set to zero.

**Parameters:**

*size* The size of the handle in bytes that will be allocated.

**Returns:**

A new [t\\_handle](#).

**See also:**

[sysmem\\_newhandle\(\)](#)

#### 26.21.4.15 `t_ptr sysmem_newptr (long 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:**

*size* The amount of memory to allocate.

**Returns:**

A pointer to the allocated memory, or NULL if the allocation fails.

**26.21.4.16 t\_ptr sysmem\_newptrclear (long 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:**

*size* The amount of memory to allocate.

**Returns:**

A pointer to the allocated memory, or NULL if the allocation fails.

**26.21.4.17 long sysmem\_nullterminatehandle (t\_handle h)**

Add a null terminator to a handle.

**Parameters:**

*h* A handle to null terminate.

**Returns:**

An error code.

**26.21.4.18 long sysmem\_ptrandhand (void \*p, t\_handle h, long 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:**

*p* The existing pointer whose data will be copied into the resized handle.

*h* The handle which will be enlarged by the size of the pointer.

*size* The size in bytes that will be added to the handle.

**Returns:**

The number of bytes allocated to the specified handle.

**26.21.4.19 long sysmem\_ptrbeforehand (void \*p, t\_handle h, unsigned long 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:**

*p* The existing pointer whose data will be copied into the resized handle.

*h* The handle which will be enlarged by the size of the pointer.

*size* The size in bytes that will be added to the handle.

**Returns:**

An error code.

**26.21.4.20 long sysmem\_ptrsize (void \* *ptr*)**

Find the size of a pointer. This function is similar to `_msize()`.

**Parameters:**

*ptr* The pointer whose size will be queried

**Returns:**

The number of bytes allocated to the pointer specified.

**26.21.4.21 long sysmem\_resizehandle (t\_handle *handle*, long *newsize*)**

Resize an existing handle. This function is similar to `SetHandleSize()`. It resizes an existing handle to the size specified.

**Parameters:**

*handle* The handle that will be resized.

*newsize* The new size of the handle in bytes.

**Returns:**

The number of bytes allocated to the specified handle.

**26.21.4.22 t\_ptr sysmem\_resizeptr (void \* *ptr*, long *newsize*)**

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:**

*ptr* The pointer to the memory that will be resized.

*newsize* The new size of the pointer in bytes.

**Returns:**

A pointer to the resized memory, or NULL if the allocation fails.

**26.21.4.23 t\_ptr sysmem\_resizeptrclear (void \* *ptr*, long *newsize*)**

Resize an existing pointer and clear it.

**Parameters:**

*ptr* The pointer to the memory that will be resized.

*newsize* The new size of the pointer in bytes.

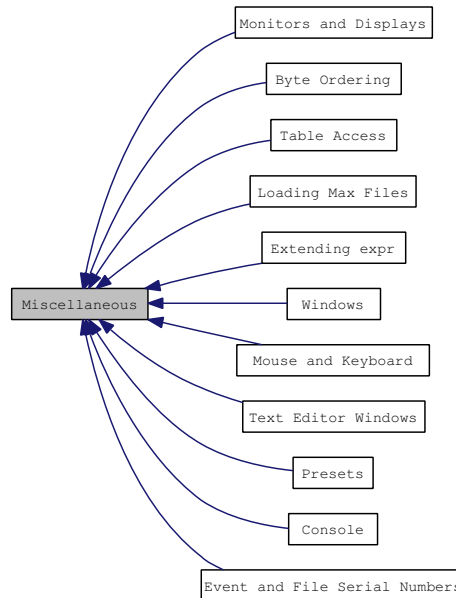
**Returns:**

A pointer to the resized memory, or NULL if the allocation fails.



## 26.22 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.*

- [Event and File Serial Numbers](#)

*If you call `outlet_int()`, `outlet_float()`, `outlet_list()`, or `outlet_anything()` inside a Qelem or during some idle or interrupt time, you should increment Max’s Event Serial Number beforehand.*

- [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](#)

## Defines

- `#define InRange(v, lo, hi) ((v)<=(hi)&&(v)>=(lo))`  
*If a value is within the specified range, then return true.*
- `#define MAX(a, b) ((a)>(b)?(a):(b))`  
*Return the higher of two values.*
- `#define MIN(a, b) ((a)<(b)?(a):(b))`  
*Return the lower of two values.*
- `#define CLIP(a, lo, hi) ( (a)>(lo)?( (a)<(hi)?(a):(hi) ):(lo) )`  
*Limit values to within a specified range.*
- `#define calcoffset(x, y) ((long)(&(((x *)0L)->y)))`  
*Find byte offset of a named member of a struct, relative to the beginning of that struct.*
- `#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](#).*

## Enumerations

- `enum e\_max\_errorcodes {`  
`MAX\_ERR\_NONE = 0,`  
`MAX\_ERR\_GENERIC = -1,`  
`MAX\_ERR\_INVALID\_PTR = -2,`  
`MAX\_ERR\_DUPLICATE = -3,`  
`MAX\_ERR\_OUT\_OF\_MEM = -4 }`  
*Standard values returned by function calls with a return type of [t\\_max\\_err](#).*
- `enum e\_max\_wind\_advise\_result {`  
`aaYes = 1,`  
`aaNo,`  
`aaCancel }`  
*Returned values from [wind\\_advise\(\)](#).*

## Functions

- void \* [globalsymbol\\_reference](#) (t\_object \*x, char \*name, char \*classname)  
*Get a reference to an object that is bound to a [t\\_symbol](#).*
- void [globalsymbol\\_dereference](#) (t\_object \*x, char \*name, 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, char \*name, long flags)  
*Bind an object to a [t\\_symbol](#).*
- void [globalsymbol\\_unbind](#) (t\_object \*x, char \*name, long flags)  
*Remove an object from being bound to a [t\\_symbol](#).*
- long [method\\_true](#) (void \*x)  
*A method that always returns true.*
- long [method\\_false](#) (void \*x)  
*A method that always returns false.*
- t\_symbol \* [symbol\\_unique](#) ()  
*Generates a unique [t\\_symbol](#) \*.*
- 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.*

- char \* [strcpy\\_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 [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().*
- short [wind\\_advise](#) (t\_object \*w, char \*s,...)  
*Throw a dialog which may have text and up to three buttons.*
- void [wind\\_setcursor](#) (short which)  
*Change the cursor.*

## 26.22.1 Define Documentation

### 26.22.1.1 #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.

Definition at line 29 of file ext\_prefix.h.

### 26.22.1.2 #define calcoffset(x, y) ((long)((x \*)0L->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 long integer representing the number of bytes into the struct where the member begins.

Definition at line 88 of file ext\_obex.h.

### 26.22.1.3 #define CLIP(a, lo, hi) ( (a)>(lo)?( (a)<(hi)?(a):(hi) ):(lo) )

Limit values to within a specified range.

#### Parameters:

- a* The value to constrain.
- lo* The low bound for the range.

*hi* The high bound for the range.

**Returns:**

Returns the value *a* constrained to the range specified by *lo* and *hi*.

Definition at line 53 of file `ext_common.h`.

**26.22.1.4 #define InRange(*v*, *lo*, *hi*) ((*v*)<=(*hi*)&&(*v*)>=(*lo*))**

If a value is within the specified range, then return true. Otherwise return false.

**Parameters:**

*v* The value to test.

*lo* The low bound for the range.

*hi* The high bound for the range.

**Returns:**

Returns true if within range, otherwise false.

Definition at line 15 of file `ext_common.h`.

**26.22.1.5 #define MAX(*a*, *b*) ((*a*)>(*b*)?(*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*.

Definition at line 27 of file `ext_common.h`.

**26.22.1.6 #define MIN(*a*, *b*) ((*a*)<(*b*)?(*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*.

Definition at line 40 of file `ext_common.h`.

## 26.22.2 Enumeration Type Documentation

### 26.22.2.1 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.

Definition at line 52 of file ext\_obex.h.

### 26.22.2.2 enum e\_max\_wind\_advise\_result

Returned values from [wind\\_advise\(\)](#).

#### Enumerator:

*aaYes* Yes button was choosen.  
*aaNo* No button was choosen.  
*aaCancel* Cancel button was choosen.

Definition at line 15 of file ext\_wind.h.

## 26.22.3 Function Documentation

### 26.22.3.1 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:

```
addresss((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);
```

**26.22.3.2 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

**26.22.3.3 void error\_unsubscribe (t\_object \* *x*)**

Remove an object as an error message recipient.

**Parameters:**

- x* The object to unsubscribe.

**26.22.3.4 t\_max\_err globalsymbol\_bind (t\_object \* *x*, char \* *name*, long *flags*)**

Bind an object to a [t\\_symbol](#).

**Parameters:**

- x* The object to bind to the [t\\_symbol](#).
- name* The name of the [t\\_symbol](#) to which the object will be bound.
- flags* Pass 0.

**Returns:**

- A Max error code.

**26.22.3.5 void globalsymbol\_dereference (t\_object \* *x*, char \* *name*, 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\(\)](#)

### 26.22.3.6 void\* globalsymbol\_reference (t\_object \*x, char \*name, char \*classname)

Get a reference to an object that is bound to a [t\\_symbol](#).

#### 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.

#### 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 referenceing another buffer, we should not l
onger reference it.
        globalsymbol_dereference((t_object *)x, x->buffer_name->s_name, "buff
er~");

        x->buffer_name = s;
    }
}
```

### 26.22.3.7 void globalsymbol\_unbind (t\_object \*x, char \*name, long flags)

Remove an object from being bound to a [t\\_symbol](#).

#### Parameters:

- x* The object from which to unbind the [t\\_symbol](#).
- name* The name of the [t\\_symbol](#) from which the object will be unbound.
- flags* Pass 0.

### 26.22.3.8 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.

**26.22.3.9 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.

**26.22.3.10 void post\_sym (void \* *x*, t\_symbol \* *s*)**

Posts a message to the Max window. This function is interrupt safe.

**Parameters:**

- x* The object's pointer
- s* Symbol to be posted in the Max window

**26.22.3.11 void quittask\_install (method *m*, void \* *a*)**

Register a function that will be called when Max exits.

**Parameters:**

- m* A function that will be called on Max exit.
- a* Argument to be used with method *m*.

**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 persistence 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.

**26.22.3.12 void quittask\_remove (method *m*)**

Unregister a function previously registered with [quittask\\_install\(\)](#).

**Parameters:**

- m* Function to be removed as a shutdown method.

### 26.22.3.13 `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:

*buffer* The destination string (already allocated) for the copy.

*count* The number of chars allocated to the buffer string.

*format* 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.

### 26.22.3.14 `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:

*dst* The destination string onto whose end the *src* string will be appended.

*src* The source string that will be copied.

*size* The number of chars allocated to the *dst* string.

### 26.22.3.15 `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:

*dst* The destination string (already allocated) for the copy.

*src* The source string that will be copied.

*size* The number of chars allocated to the *dst* string.

Referenced by `db_open()`, and `db_query_table_new()`.

### 26.22.3.16 `t_symbol* symbol_unique ()`

Generates a unique `t_symbol *`. The symbol will be formatted somewhat like "u123456789".

#### Returns:

This function returns a unique `t_symbol *`.

**26.22.3.17 t\_max\_err symbolarray\_sort (long *ac*, t\_symbol \*\* *av*)**

Performs an ASCII sort on an array of [t\\_symbol](#) \*s.

**Parameters:**

- ac* The count of [t\\_symbol](#) \*s in *av*
- av* An array of [t\\_symbol](#) \*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.

**26.22.3.18 short wind\_advise (t\_object \* *w*, 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:**

- w* The window with which this dialog is associated.
- s* A string with any `sprintf()`-like formatting to be displayed.
- ... Any variables that should be substituted in the string defined by *s*.

**Returns:**

One of the values defined in [e\\_max\\_wind\\_advise\\_result](#), depending on what the user selected.

**26.22.3.19 void wind\_setcursor (short *which*)**

Change the cursor.

**Parameters:**

*which* One of the following predefined cursors:

```
#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
```

**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.

## 26.23 Console

Collaboration diagram for Console:



### Functions

- void `post` (char \*fmt,...)  
*Print text to the Max window.*
- void `cpost` (char \*fmt,...)  
*Print text to the system console.*
- void `error` (char \*fmt,...)  
*Print an error to the Max window.*
- void `ouchstring` (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, char \*s,...)  
*Print text to the Max window, linked to an instance of your object.*
- void `object_error` (t\_object \*x, 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, 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, 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.*

### 26.23.1 Function Documentation

#### 26.23.1.1 void `cpost` (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:**

*fmt* 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:**

Particularly on MacOS 10.5, posting to Console.app can be a computationally expensive operation. Use with care.

**See also:**

[post\(\)](#)  
[object\\_post\(\)](#)

Referenced by [db\\_query\(\)](#).

**26.23.1.2 void error (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\(\)](#), though 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:**

*fmt* 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\(\)](#)  
[post\(\)](#)  
[cpost\(\)](#)

**26.23.1.3 void object\_error (t\_object \**x*, 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:**

*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\\_warn\(\)](#)

#### 26.23.1.4 void `object_error_obtrusive` (`t_object *x`, `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\(\)](#)

#### 26.23.1.5 void `object_post` (`t_object *x`, `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:**

*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`.

**Remarks:**

Example:

```
void myMethod(myObject *x, long someArgument)
{
    object_post((t_object*)x, "This is my argument: %ld", someArgument);
}
```

**See also:**

[object\\_error\(\)](#)

### 26.23.1.6 void object\_warn (t\_object \*x, 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:

- 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\(\)](#)

### 26.23.1.7 void ouchstring (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:

- 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:

[error\(\)](#)  
[object\\_error\(\)](#)  
[object\\_error\\_obtrusive\(\)](#)

### 26.23.1.8 void post (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:

- fmt* 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:**

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\(\)](#)

**26.23.1.9 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:**

*ap* The address of a [t\\_atom](#) to print.

**See also:**

[object\\_post\(\)](#)  
[post\(\)](#)  
[cpost\(\)](#)



## 26.24 Byte Ordering

Utilities for swapping the order of bytes to match the Endianness of the required platform.

Collaboration diagram for Byte Ordering:



### Defines

- `#define C74_LITTLE_ENDIAN 1`  
*A macro that indicates whether or not the current architecture uses Little-endian byte ordering (such as is used on an i386 processor).*
- `#define C74_BIG_ENDIAN 0`  
*A macro that indicates whether or not the current architecture uses Big-endian byte ordering (such as is used on a PPC processor).*
- `#define BYTEORDER_SWAPW16(x) (((short)((((unsigned short)(x))>>8)&0x00ff)+((((unsigned short)(x))<<8)&0xff00)))`  
*Switch the byte ordering of a short integer.*
- `#define BYTEORDER_SWAPW32(x)`  
*Switch the byte ordering of an integer.*
- `#define BYTEORDER_SWAPF32 byteorder_swapf32`  
*Switch the byte ordering of a float.*
- `#define BYTEORDER_SWAPF64 byteorder_swapf64`  
*Switch the byte ordering of a double.*
- `#define BYTEORDER_LSBW16(x) (x)`  
*Switch the byte ordering of a short integer from the native swapping to Little-endian (Least Significant Byte).*
- `#define BYTEORDER_LSBW32(x) (x)`  
*Switch the byte ordering of an integer from the native swapping to Little-endian (Least Significant Byte).*
- `#define BYTEORDER_LSBF32(x) (x)`  
*Switch the byte ordering of a float from the native swapping to Little-endian (Least Significant Byte).*
- `#define BYTEORDER_LSBF64(x) (x)`  
*Switch the byte ordering of a double from the native swapping to Little-endian (Least Significant Byte).*
- `#define BYTEORDER_MSBW16(x) BYTEORDER_SWAPW16(x)`  
*Switch the byte ordering of a short integer from the native swapping to Big-endian (Most Significant Byte).*
- `#define BYTEORDER_MSBW32(x) BYTEORDER_SWAPW32(x)`  
*Switch the byte ordering of an integer from the native swapping to Big-endian (Most Significant Byte).*

- `#define BYTEORDER_MSBF32(x) BYTEORDER_SWAPF32(x)`  
*Switch the byte ordering of a float from the native swapping to Big-endian (Most Significant Byte).*
- `#define BYTEORDER_MSBF64(x) BYTEORDER_SWAPF64(x)`  
*Switch the byte ordering of a double from the native swapping to Big-endian (Most Significant Byte).*

### 26.24.1 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.

### 26.24.2 Define Documentation

#### 26.24.2.1 `#define BYTEORDER_LSBF32(x) (x)`

Switch the byte ordering of a float from the native swapping to Little-endian (Least Significant Byte). If the current environment is already Little-endian, then the returned value is not byteswapped.

##### Parameters:

*x* A float.

##### Returns:

A float with the byte-ordering swapped if neccessary.

Definition at line 111 of file ext\_byteorder.h.

#### 26.24.2.2 `#define BYTEORDER_LSBF64(x) (x)`

Switch the byte ordering of a double from the native swapping to Little-endian (Least Significant Byte). If the current environment is already Little-endian, then the returned value is not byteswapped.

##### Parameters:

*x* A double.

##### Returns:

A double with the byte-ordering swapped if neccessary.

Definition at line 121 of file ext\_byteorder.h.

**26.24.2.3 #define BYTEORDER\_LSBW16(x) (x)**

Switch the byte ordering of a short integer from the native swapping to Little-endian (Least Significant Byte). If the current environment is already Little-endian, then the returned value is not byteswapped.

**Parameters:**

*x* A short integer.

**Returns:**

A short integer with the byte-ordering swapped if neccessary.

Definition at line 91 of file ext\_byteorder.h.

**26.24.2.4 #define BYTEORDER\_LSBW32(x) (x)**

Switch the byte ordering of an integer from the native swapping to Little-endian (Least Significant Byte). If the current environment is already Little-endian, then the returned value is not byteswapped.

**Parameters:**

*x* An integer.

**Returns:**

An integer with the byte-ordering swapped if neccessary.

Definition at line 101 of file ext\_byteorder.h.

**26.24.2.5 #define BYTEORDER\_MSBF32(x) BYTEORDER\_SWAPF32(x)**

Switch the byte ordering of a float from the native swapping to Big-endian (Most Significant Byte). If the current environment is already Big-endian, then the returned value is not byteswapped.

**Parameters:**

*x* A float.

**Returns:**

A float with the byte-ordering swapped if neccessary.

Definition at line 151 of file ext\_byteorder.h.

**26.24.2.6 #define BYTEORDER\_MSBF64(x) BYTEORDER\_SWAPF64(x)**

Switch the byte ordering of a double from the native swapping to Big-endian (Most Significant Byte). If the current environment is already Big-endian, then the returned value is not byteswapped.

**Parameters:**

*x* A double.

**Returns:**

A double with the byte-ordering swapped if neccessary.

Definition at line 161 of file ext\_byteorder.h.

**26.24.2.7 #define BYTEORDER\_MSBW16(x) BYTEORDER\_SWAPW16(x)**

Switch the byte ordering of a short integer from the native swapping to Big-endian (Most Significant Byte). If the current environment is already Big-endian, then the returned value is not byteswapped.

**Parameters:**

*x* A short integer.

**Returns:**

A short integer with the byte-ordering swapped if neccessary.

Definition at line 131 of file ext\_byteorder.h.

**26.24.2.8 #define BYTEORDER\_MSBW32(x) BYTEORDER\_SWAPW32(x)**

Switch the byte ordering of an integer from the native swapping to Big-endian (Most Significant Byte). If the current environment is already Big-endian, then the returned value is not byteswapped.

**Parameters:**

*x* An integer.

**Returns:**

An integer with the byte-ordering swapped if neccessary.

Definition at line 141 of file ext\_byteorder.h.

**26.24.2.9 #define BYTEORDER\_SWAPF32 byteorder\_swapf32**

Switch the byte ordering of a float.

**Parameters:**

*x* A float.

**Returns:**

A float with the byte-ordering swapped.

Definition at line 56 of file ext\_byteorder.h.

**26.24.2.10 #define BYTEORDER\_SWAPF64 byteorder\_swapf64**

Switch the byte ordering of a double.

**Parameters:**

*x* A double.

**Returns:**

A double.

Definition at line 64 of file ext\_byteorder.h.

**26.24.2.11 #define BYTEORDER\_SWAPW16(x) (((short)((((unsigned short)(x))>>8)&0x00ff)+((((unsigned short)(x))<<8)&0xff00)))**

Switch the byte ordering of a short integer.

**Parameters:**

*x* A short integer.

**Returns:**

A short integer with the byte-ordering swapped.

Definition at line 40 of file ext\_byteorder.h.

**26.24.2.12 #define BYTEORDER\_SWAPW32(x)****Value:**

```
((long) (((((unsigned long) (x))>>24L)&0x000000ff)+(((unsigned long) (x))>>8L)&0x0000ff00)+ \
          (((((unsigned long) (x))<<24L)&0xff000000)+(((unsigned long) (x))<<8L)&0x00ff0000)))
```

Switch the byte ordering of an integer.

**Parameters:**

*x* An integer.

**Returns:**

An integer with the byte-ordering swapped.

Definition at line 48 of file ext\_byteorder.h.

**26.24.2.13 #define C74\_BIG\_ENDIAN 0**

A macro that indicates whether or not the current architecture uses Big-endian byte ordering (such as is used on a PPC processor). Note that this macro is always defined; it will be either a 0 or a 1.

Definition at line 29 of file ext\_byteorder.h.

**26.24.2.14 #define C74\_LITTLE\_ENDIAN 1**

A macro that indicates whether or not the current architecture uses Little-endian byte ordering (such as is used on an i386 processor). Note that this macro is always defined; it will be either a 0 or a 1.

Definition at line 21 of file ext\_byteorder.h.

## 26.25 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 `Ex_ex`  
*ex\_ex.*
- struct `t_expr`  
*Struct for an instance of expr.*

### Defines

- #define `ex_int` `ex_cont.v_int`  
*shortcut for accessing members of an `Ex_ex` struct’s `ex_cont` union.*
- #define `exflt` `ex_cont.vflt`  
*shortcut for accessing members of an `Ex_ex` struct’s `ex_cont` union.*
- #define `exop` `ex_cont.op`  
*shortcut for accessing members of an `Ex_ex` struct’s `ex_cont` union.*
- #define `exptr` `ex_cont.ptr`  
*shortcut for accessing members of an `Ex_ex` struct’s `ex_cont` union.*

### Enumerations

- enum `e_max_expr_types` {  
`ET_INT` = 0x1,  
`ET_FLT` = 0x2,  
`ET_OP` = 0x3,  
`ET_STR` = 0x4,  
`ET_TBL` = 0x5,  
`ET_FUNC` = 0x6,  
`ET_SYM` = 0x7,  
`ET_VSYM` = 0x8,  
`ET_LP` = 0x9,  
`ET_LB` = 0x10,

```

ET_II = 0x11,
ET_FI = 0x12,
ET_SI = 0x13 }
Defines for ex_type.

```

## Functions

- void \* **expr\_new** (short argc, **t\_atom** \*argv, **t\_atom** \*types)  
*Create a new expr object.*
- short **expr\_eval** (**t\_expr** \*x, short argc, **t\_atom** \*argv, **t\_atom** \*result)  
*Evaluate an expression in an expr object.*

### 26.25.1 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.

### 26.25.2 Enumeration Type Documentation

#### 26.25.2.1 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

```

Definition at line 59 of file ext\_expr.h.



### 26.25.3 Function Documentation

#### 26.25.3.1 short expr\_eval (t\_expr \* x, short 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 Atoms that will be substituted for variable arguments (such as \$i1) in the expression. Unused arguments should be of type [A\\_NOTHING](#).
- result* A pre-existing 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 Atoms. 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](#).

#### 26.25.3.2 void\* expr\_new (short argc, t\_atom \* argv, t\_atom \* types)

Create a new expr object.

**Parameters:**

- argc* Count of arguments in argv.
- argv* Arguments that are used to create the expr. See the example below for details.
- types* A pre-existing array of nine t\_atoms, that will hold the types of any variable arguments created in the expr. The types are returned in the a\_type field of each [t\\_atom](#). If an argument was not present, [A\\_NOTHING](#) 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 Atoms, 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)
+ (A_SYM)
$f3 (A_SYM)
+ (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

## 26.26 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.*

### 26.26.1 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`.”

### 26.26.2 Function Documentation

#### 26.26.2.1 short `table_dirty` (`t_symbol *s`)

Mark a table object as having changed data.

##### Parameters:

- `s` Symbol containing the name of a table object.

##### Returns:

If no table is associated with `tableName`, `table_dirty` returns a non-zero result.

#### 26.26.2.2 short `table_get` (`t_symbol *s`, long `***hp`, long `*sp`)

Get a handle to the data in a named table object.

##### Parameters:

- `s` Symbol containing the name of the table object to find.
- `hp` Address of a handle where the table’s data will be returned if the named table object is found.
- `sp` 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);
}
```

## 26.27 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](#)

## 26.28 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, long val)  
*Restore the state of your object with a set message.*
- void `preset_int` (void \*x, long n)  
*Restore the state of your object with an int message.*

### 26.28.1 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);
    }
}
```

## 26.28.2 Function Documentation

### 26.28.2.1 void preset\_int (void \* x, 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.

### 26.28.2.2 void preset\_set (t\_object \* obj, 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.

### 26.28.2.3 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:

- fmt* 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.



## 26.29 Event and File Serial Numbers

If you call `outlet_int()`, `outlet_float()`, `outlet_list()`, or `outlet_anything()` inside a Qelem or during some idle or interrupt time, you should increment Max's Event Serial Number beforehand.

Collaboration diagram for Event and File Serial Numbers:



### Functions

- void `evnum_incr` (void)  
*Increment the event serial number.*
- long `evnum_get` (void)  
*Get the current value of the event serial number.*
- long `serialno` (void)  
*Get a unique number for each Patcher file saved.*

### 26.29.1 Detailed Description

If you call `outlet_int()`, `outlet_float()`, `outlet_list()`, or `outlet_anything()` inside a Qelem or during some idle or interrupt time, you should increment Max's Event Serial Number beforehand. This number can be read by objects that want to know if two messages they have received occurred at the same logical "time" (in response to the same event). Max increments the serial number for each tick of the clock, each key press, mouse click, and MIDI event. Note that this is different from the file serial number returned by the `serialno()` function. The file serial number is only incremented when patchers are saved in files. If more than one patcher is saved in a file, the file serial number will change but the event serial number will not.

### 26.29.2 Using Event Serial Numbers

Here is a Max patch that includes an object called `simul` that would use the information returned by `evnum_get` to return a 1 if the right and left inlets receive messages at the same time, 0 if not. The number boxes below show the results of clicking on the button objects or typing a key.

### 26.29.3 Function Documentation

#### 26.29.3.1 long `evnum_get` (void)

Get the current value of the event serial number.

#### Returns:

The current value of the event serial number.

**26.29.3.2 long serialno (void)**

Get a unique number for each Patcher file saved. This function returns a serial number that is incremented each time a Patcher file is saved. This routine is useful for objects like table and coll that have multiple objects that refer to the same data, and can “embed” the data inside a Patcher file. If the serial number hasn’t changed since your object was last saved, you can detect this and avoid saving multiple copies of the object’s data.

**Returns:**

The serial number.

## 26.30 Loading Max Files

Several high-level functions permit you to load patcher files.

Collaboration diagram for Loading Max Files:



### Functions

- short [readtohandle](#) (char \*name, short volume, char \*\*\*h, long \*sizep)  
*Load a data file into a handle.*
- void \* [fileload](#) (char \*name, short vol)  
*Load a patcher file by name and volume reference number.*
- void \* [intload](#) (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](#) (char \*name)  
*Load a patcher file located in the Max search path by name.*

### 26.30.1 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.

### 26.30.2 Function Documentation

#### 26.30.2.1 void\* fileload (char \* name, short vol)

Load a patcher file by name and volume reference number.

##### Parameters:

**name** Filename of the patcher file to load (C string).

**vol** 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.

### 26.30.2.2 void\* intload (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()` is 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:

*name* Name of the file to open.

*volume* Path ID specifying the location of the file.

*s* A symbol.

*ac* Count of t\_atoms in av. To properly open a patcher file, ac should be 9.

*av* Array of t\_atoms that will replace the changeable arguments 1-9. The default behavior could be to set all these to t\_atoms of type `A_LONG` with a value of 0.

*couldedit* 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 just 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.

### 26.30.2.3 short readtohandle (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:

*name* Name of the patcher file to load.

*volume* Path ID specifying the location of the file.

*h* Pointer to a handle variable that will receive the handle that contains the data in the file.

*sizep* Size of the handle returned in h.

#### 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 *sizep*. `readtohandle` returns 0 if the file was opened and read successfully, and non-zero if there was an error.

### 26.30.2.4 void\* stringload (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:**

*name* Filename of the patcher file to load (C string).

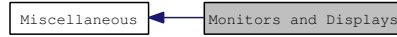
**Returns:**

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.

## 26.31 Monitors and Displays

Functions for finding our information about the environment.

Collaboration diagram for Monitors and Displays:



### Functions

- long [jmonitor\\_getnumdisplays](#) ()  
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.

### 26.31.1 Detailed Description

Functions for finding our information about the environment.

### 26.31.2 Function Documentation

#### 26.31.2.1 void [jmonitor\\_getdisplayrect](#) (long workarea, long displayindex, [t\\_rect](#) \* rect)

Return the [t\\_rect](#) for a given display.

##### Parameters:

- workarea** Set workarea non-zero to clip out things like dock / task bar.
- displayindex** The index number for a monitor. The primary monitor has an index of 0.
- rect** The address of a valid [t\\_rect](#) whose values will be filled-in upon return.

#### 26.31.2.2 void [jmonitor\\_getdisplayrect\\_foralldisplays](#) (long workarea, [t\\_rect](#) \* rect)

Return a union of all display rects.

##### Parameters:

- workarea** Set workarea non-zero to clip out things like dock / task bar.
- rect** The address of a valid [t\\_rect](#) whose values will be filled-in upon return.

**26.31.2.3 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:**

*workarea* Set workarea non-zero to clip out things like dock / task bar.

*pt* A point, for which the monitor will be determined and the rect returned.

*rect* The address of a valid t\_rect whose values will be filled-in upon return.

**26.31.2.4 long jmonitor\_getnumdisplays ()**

Return the number of monitors on which can be displayed.

**Returns:**

The number of monitors.

## 26.32 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.*

### 26.32.1 Function Documentation

#### 26.32.1.1 `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.

#### 26.32.1.2 `t_object* jwind_getat` (long index)

Return a pointer to the window with a given index.

##### Parameters:

*index* Get window at index (0 to count-1).

##### Returns:

A pointer to a window object.

#### 26.32.1.3 `long jwind_getcount` (void)

Determine how many windows exist.

##### Returns:

The number of windows.



## 26.33 Mouse and Keyboard

Collaboration diagram for Mouse and Keyboard:



### Enumerations

- enum `t_modifiers` {  
`eCommandKey` = 1,  
`eShiftKey` = 2,  
`eControlKey` = 4,  
`eAltKey` = 8,  
`eLeftButton` = 16,  
`eRightButton` = 32,  
`eMiddleButton` = 64,  
`ePopupMenu` = 128,  
`eCapsLock` = 256,  
`eAutoRepeat` = 512 }

*Bit mask values for various meta-key presses on the keyboard.*

- 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.*

## Functions

- [t\\_modifiers\\_jkeyboard\\_getcurrentmodifiers](#) ()  
*Return the last known combination of modifier keys being held by the user.*
- [t\\_modifiers\\_jkeyboard\\_getcurrentmodifiers\\_realtime](#) ()  
*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.*

## 26.33.1 Enumeration Type Documentation

### 26.33.1.1 enum t\_jmouse\_cursortype

Mouse cursor types.

#### Enumerator:

**JMOUSE\_CURSOR\_NONE** None.  
**JMOUSE\_CURSOR\_ARROW** Arrow.  
**JMOUSE\_CURSOR\_WAIT** Wait.  
**JMOUSE\_CURSOR\_IBEAM** I-Beam.  
**JMOUSE\_CURSOR\_CROSSHAIR** Crosshair.  
**JMOUSE\_CURSOR\_COPYING** Copying.  
**JMOUSE\_CURSOR\_POINTINGHAND** Pointing Hand.  
**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.

Definition at line 1944 of file jpatcher\_api.h.

### 26.33.1.2 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.

Definition at line 1825 of file jpatcher\_api.h.

## 26.33.2 Function Documentation

### 26.33.2.1 t\_modifiers jkeyboard\_getcurrentmodifiers ()

Return the last known combination of modifier keys being held by the user.

#### Returns:

The current modifier keys that are activated.

### 26.33.2.2 t\_modifiers jkeyboard\_getcurrentmodifiers\_realtime ()

Return the current combination of modifier keys being held by the user.

#### Returns:

The current modifier keys that are activated.

**26.33.2.3 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.

**26.33.2.4 void jmouse\_setcursor (t\_object \* patcherview, t\_object \* box, t\_jmouse\_cursortype type)**

Set the mouse cursor.

**Parameters:**

- patcherview* The patcherview for which the cursor should be applied.
- box* The box for which the cursor should be applied.
- type* The type of cursor for the mouse to use.

**26.33.2.5 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:**

- patcherview* The patcherview containing the box upon which the mouse coordinates are based.
- box* The box upon which the mouse coordinates are based.
- bx* The new x-coordinate of the mouse cursor position.
- by* The new y-coordinate of the mouse cursor position.

**26.33.2.6 void jmouse\_setposition\_global (int x, int y)**

Set the position of the mouse cursor in screen coordinates.

**Parameters:**

- x* The new x-coordinate of the mouse cursor position.
- y* The new y-coordinate of the mouse cursor position.

**26.33.2.7 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:**

- patcherview* The patcherview upon which the mouse coordinates are based.
- cx* The new x-coordinate of the mouse cursor position.
- cy* The new y-coordinate of the mouse cursor position.

## 26.34 Example Projects

### Files

- file [attrtester.c](#)  
*attrtester - a max object shell jeremy bernstein - [jeremy@bootsquad.com](mailto:jeremy@bootsquad.com)*
- file [buddy.c](#)  
*buddy.c --- data flow management object*
- file [collect.cpp](#)  
*collect - collect numbers and operate on them.*
- file [dbcuelist.c](#)  
*dbcuelist - demonstrate use of a sqlite database*
- file [dbviewer.c](#)  
*dbviewer - demonstrate use of database views for sqlite and jdataview*
- file [delay2.c](#)  
*delay2 - an ITM-based delay*
- file [dspstress~.c](#)  
*dspstress~ - very simple msp object that does nothing except eat up a specified % of processor time*
- file [dummy.c](#)  
*dummy - a dummy object jeremy bernstein - [jeremy@bootsquad.com](mailto:jeremy@bootsquad.com)*
- file [fftinfo~.c](#)  
*fftinfo~.c - communicate fft size and hop, etc.*
- file [filebyte.c](#)  
*filebyte - similar to filein.*
- file [filedate.c](#)  
*filedate - manage search paths Accesses a file on disk and outputs a given byte of the file's data.*
- file [filein.c](#)  
*filein - read in a file of binary data and output the data at various points in the file*
- file [folder.c](#)  
*folder - list the files in a specific folder*
- file [index~.c](#)  
*index~ - SDK example of an object which accesses an MSP buffer~*
- file [iter.c](#)  
*iter - sequential list unpacking*
- file [iterate2.c](#)

*iterate2* - object that iterates through a patcher and its subpatchers *jeremy bernstein* - [jeremy@bootsquad.com](mailto:jeremy@bootsquad.com)

- file [iterator.c](#)

*iterator* - patch iterator *jeremy bernstein* - [jeremy@bootsquad.com](mailto:jeremy@bootsquad.com)

- file [lores~.c](#)

*lores* - A low pass controllable with *freq* and *res*

- file [match.c](#)

*match* - triggering based on sequential input

- file [minimum.c](#)

*minimum* - output the minimum of a group of numbers

- file [past.c](#)

*past* - triggering after numbers get "past" a certain point

- file [pictmeter~.c](#)

*pictmeter~* - audio meter that works by resizing an image

- file [plussz.c](#)

*plussz.c* - one of the simplest max objects you can make - rdd 2001 (*plussz* is/was the name of a Hungarian vitamin C tablet-drink from the early 90s)

- file [plussz2.c](#)

*plussz2.c* - a version of *plussz2* that demonstrates the use of proxy inlets.

- file [plussz~.c](#)

*plussz~* - a very simple example of a basic MSP object

- file [scripto.c](#)

*scripto* - patcher scripting from C example *scripto* makes a custom UI object and then puts it in a window -- so it is similar to the old "kalim" example

- file [simplejs.c](#)

*simplejs* - a max object used as a JavaScript class

- file [simplemax.c](#)

*simplemax* - a max object shell *jeremy bernstein* - [jeremy@bootsquad.com](mailto:jeremy@bootsquad.com)

- file [simplemsp~.c](#)

*simplemsp* - an MSP object shell *jeremy bernstein* - [jeremy@bootsquad.com](mailto:jeremy@bootsquad.com)

- file [simpletext.c](#)

*simpletext* - show use of text reading and editing

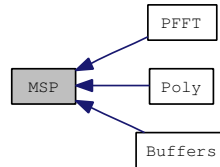
- file [simpwave~.c](#)

*simpwave~* - a simple wavetable oscillator using *buffer~*

- file [thresh.c](#)  
*thresh - forming lists*
- file [times~.c](#)  
*times~ - the \*~ signal operator SDK example to illustrate platform-safe AltiVec optimization*
- file [uisimp2.c](#)  
*uisimp - a very simple ui object - step 4*
- file [uisimp3.c](#)  
*uisimp - a very simple ui object - step 5*
- file [uisimp4.c](#)  
*uisimp - a very simple ui object - step 6*
- file [uisimp5.c](#)  
*uisimp - a very simple ui object - step 7*
- file [uitester.c](#)  
*uitester - demonstrate the drawing of various objects using jgraphics*
- file [uitextfield.c](#)  
*uitextfield - demonstrate the textfield with keyboard input*
- file [urner.c](#)  
*urner - a max object shell jeremy bernstein - [jeremy@bootsquad.com](mailto:jeremy@bootsquad.com)*
- file [whosyourdaddy.c](#)  
*whosyourdaddy - who's the parent patcher jeremy bernstein - [jeremy@bootsquad.com](mailto:jeremy@bootsquad.com)*
- file [windowwatcher.c](#)  
*windowwatcher - Demonstrate how to get notifications about the window for a patcher in which an object exists.*

## 26.35 MSP

Collaboration diagram for MSP:



### Data Structures

- 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.*

### 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.*

### Defines

- #define [Z\\_NO\\_INPLACE](#) 1  
*flag indicating the object doesn't want signals in place*
- #define [Z\\_PUT\\_LAST](#) 2  
*when list of ugens is resorted, put this object at end*
- #define [Z\\_PUT\\_FIRST](#) 4  
*when list of ugens is resorted, put this object at beginning*
- #define [PI](#) 3.14159265358979323846



*The pi constant.*

- #define `TWOPI` 6.28318530717958647692

*Twice the pi constant.*

- #define `PIOVERTWO` 1.57079632679489661923

*Half of the pi constant.*

- #define `dsp_setup` `z_dsp_setup`

*This is commonly used rather than directly calling `z_dsp_setup()` in MSP objects.*

- #define `dsp_free` `z_dsp_free`

*This is commonly used rather than directly calling `z_dsp_free()` in MSP objects.*

## Typedefs

- typedef int `t_int`

*An integer.*

- typedef float `t_float`

*A float.*

- typedef float `t_sample`

*A sample value.*

- 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.*

## Enumerations

- enum {  
`SYS_MAXBLKSIZE` = 2048,  
`SYS_MAXSIG` = 250 }

*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.*
- void [dsp\\_add](#) (t\_perfroutine f, int n,...)  
*Call this function in your MSP object's dsp method.*
- void [dsp\\_addv](#) (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\\_dspinitbox](#) (t\_class \*c)  
*This routine must be called in your object's initialization routine.*
- void [class\\_dspinitjbox](#) (t\_class \*c)  
*Configure a class to be ready for use with the MSP signal chain.*

## 26.35.1 Define Documentation

### 26.35.1.1 #define PI 3.14159265358979323846

The pi constant.

Definition at line 164 of file z\_dsp.h.

### 26.35.1.2 #define PIOVERTWO 1.57079632679489661923

Half of the pi constant.

Definition at line 172 of file z\_dsp.h.

### 26.35.1.3 #define TWOPI 6.28318530717958647692

Twice the pi constant.

Definition at line 168 of file z\_dsp.h.

## 26.35.2 Typedef Documentation

### 26.35.2.1 typedef float t\_float

A float.

Definition at line 12 of file `z_dsp.h`.

#### 26.35.2.2 `typedef int t_int`

An integer.

Definition at line 10 of file `z_dsp.h`.

#### 26.35.2.3 `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.

Definition at line 149 of file `z_dsp.h`.

#### 26.35.2.4 `typedef float t_sample`

A sample value.

Definition at line 14 of file `z_dsp.h`.

### 26.35.3 Enumeration Type Documentation

#### 26.35.3.1 `anonymous enum`

MSP System Properties.

##### Enumerator:

`SYS_MAXBLKSIZE` a good number for a maximum signal vector size

`SYS_MAXSIGs` number of signal inlets you can have in an object

Definition at line 28 of file `z_dsp.h`.

### 26.35.4 Function Documentation

#### 26.35.4.1 `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`, `drawline`, `userconnect`, and `enable`.

This routine is for normal (non-user-interface objects). It must be called prior to calling `class_register()` for your class.

##### Parameters:

`c` The class to make dsp-ready.

##### See also:

[class\\_dspinitbox\(\)](#)

### 26.35.4.2 void class\_dspinitbox (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, drawline, userconnect, and enable.

This routine is for normal user-interface objects.

#### Parameters:

*c* The class to make dsp-ready.

#### See also:

[class\\_dspinit\(\)](#)

### 26.35.4.3 void class\_dspinitjbox (t\_class \* *c*)

Configure a class to be ready for use with the MSP signal chain. You must call this function when your class is initialized (typically in the main() function) for an object to process audio.

#### Parameters:

*c* The pointer to the class being configured.

### 26.35.4.4 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. *argc*, the number of arguments to your perform method, should be followed by *argc* additional arguments, all of which must be the size of a pointer or a long.

#### Parameters:

*f* The perform routine to use for processing audio.  
*n* The number of arguments that will follow  
... The arguments that will be passed to the perform routine.

### 26.35.4.5 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:

*f* The perform routine to use for processing audio.  
*n* The number of arguments that will follow in the vector parameter.  
*vector* The arguments that will be passed to the perform routine.

**26.35.4.6 int sys\_getblksize (void)**

Query MSP for the current global vector (block) size.

**Returns:**

The current global vector size for the MSP environment.

**26.35.4.7 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.

**26.35.4.8 int sys\_getmaxblksize (void)**

Query MSP for the maximum global vector (block) size.

**Returns:**

The maximum global vector size for the MSP environment.

**26.35.4.9 float sys\_getsr (void)**

Query MSP for the global sample rate.

**Returns:**

The global sample rate of the MSP environment.

**26.35.4.10 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](#)

#### 26.35.4.11 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:

*x* Your object's pointer.

*nsignals* The number of signal/proxy inlets to create for the object.

##### See also:

[dsp\\_setup](#)

## 26.36 Buffers

Your object can access shared data stored in an MSP `buffer~` object.

Collaboration diagram for Buffers:



### Data Structures

- struct `t_buffer`

*Data structure for the `buffer~` object.*

#### 26.36.1 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 gain access to the `t_buffer` struct. Consider the following example.

```

t_symbol *s;
t_object *o;

s = gensym("foo");
o = s->s_thing;

// if an object is bound to the symbol "foo", then o is that object.
if (ob_sym(o) == gensym("buffer~")) {
    // that object is a buffer~, so we can use it
    x->x_buffer = (t_buffer*)o;
}

```

Having stored a pointer to the `buffer~` is the first step toward working with its data. However, you must not go accessing the data directly without taking some precautions regarding thread-safety.

To access the data in a buffer you first increment the `b_inuse` member of the `t_buffer`'s struct. Then you perform the requisite operations on the data, which is stored in the `b_samples` member. When you are done you decrement the `b_inuse` member to return it to the state in which you found it.

In the past you may have set the buffer's `b_inuse` flag directly and cleared it when you were done. This is no longer good enough, and you must instead use the threadsafe macros `ATOMIC_INCREMENT` and `ATOMIC_DECREMENT` for modifying the `b_inuse` flag. The example below demonstrates what this might look like in an MSP object's perform routine. Notice that extra care has been taken to ensure that the `ATOMIC_INCREMENT` is always balanced with an `ATOMIC_DECREMENT` call.

```

ATOMIC_INCREMENT(&x->w_buf->b_inuse);
if (!x->w_buf->b_valid) {
    ATOMIC_DECREMENT(&x->w_buf->b_inuse);
    goto byebye;
}

// do something with the buffer

ATOMIC_DECREMENT(&x->w_buf->b_inuse);
byebye:
return (w + 7);

```

A class that accesses `buffer~` objects is the `simpwave~` object that is included with Max 5 SDK example projects.



## 26.37 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.*

#### 26.37.1 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~
}
```

## 26.38 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);
    }
}

```

## 26.39 Objects

More information for this section can be gleaned from the `patrsdk.pdf` in the Max 4.5.5 SDK.

### 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.*

### Defines

- #define `MAGIC` 1758379419L  
*Magic number used to determine if memory pointed to by a `t_object*` is valid.*
- #define `NOGOOD(x)` (((struct object \*)x)->o\_magic != MAGIC)  
*Returns true if a pointer is not a valid object.*
- #define `MAXARG` 7  
*Maximum number of arguments that can be passed as a typed-list rather than using `A_GIMME`.*

### Functions

- `t_object * newobject_sprintf(t_object *patcher, char *fmt,...)`  
*Create a new object in a specified patcher with values using a combination of attribute and sprintf syntax.*
- `t_object * newobject_fromdictionary(t_object *patcher, t_dictionary *d)`  
*Place a new object into a patcher.*
- `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.*
- 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.*
- [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.*
- `t_max_err object_method_parse` (`t_object *x`, `t_symbol *s`, 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`, 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, 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.*

## 26.39.1 Detailed Description

More information for this section can be gleaned from the `pattrsdk.pdf` in the Max 4.5.5 SDK.

See also:

<http://www.cycling74.com/twiki/bin/view/ProductDocumentation/JitterSdkObjectModel>  
<http://www.cycling74.com/twiki/bin/view/ProductDocumentation/JitterSdkRegNotify>

## 26.39.2 Define Documentation

### 26.39.2.1 #define MAXARG 7

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](#).

Definition at line 101 of file ext\_mess.h.

## 26.39.3 Function Documentation

### 26.39.3.1 void classname\_openhelp (char \* *classname*)

Open the help patcher for a given object class name.

#### Parameters:

*classname* The class name for which to open the help patcher.

### 26.39.3.2 void classname\_openquery (char \* *classname*)

Open a search in the file browser for files with the name of the given class.

#### Parameters:

*classname* The class name for which to query.

### 26.39.3.3 void classname\_openrefpage (char \* *classname*)

Open the reference page for a given object class name.

#### Parameters:

*classname* The class name for which to open the reference page.

### 26.39.3.4 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:

*patcher* An instance of a patcher object.

*d* 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\(\)](#)

**26.39.3.5 t\_object\* newobject\_sprintf (t\_object\* *patcher*, char\* *fmt*, ...)**

Create a new object in a specified patcher with values using a combination of attribute and sprintf syntax.

**Parameters:**

***patcher*** An instance of a patcher object.

***fmt*** 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\(\)](#)

**26.39.3.6 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:**

*c* The class pointer, returned by [class\\_new\(\)](#)

**Returns:**

This function returns a new instance of an object class if successful, or NULL if unsuccessful.

**26.39.3.7 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:**

*name\_space* The namespace of the registered object. This should be the same value used in [object\\_register\(\)](#) to register the object. If you don't know the registered object's namespace, the [object\\_findregisteredbyptr\(\)](#) function can be used to determine it.

*s* The name of the registered object in the namespace. If you don't know the name of the registered object, the [object\\_findregisteredbyptr\(\)](#) function can be used to determine it.

*x* 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\(\)](#)

### 26.39.3.8 `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:

- x* The attaching client object. Generally, this is the pointer to your Max object.
- registeredobject* 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\(\)](#)

### 26.39.3.9 `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:

- x* The attaching client object. Generally, this is the pointer to your Max object.
- object\_to\_attach* A pointer to the object to which you wish to registered and then to which to attach.
- reg\_name\_space* The namespace in which to register the *object\_to\_attach*.

#### Returns:

A Max error code.

#### See also:

[object\\_register\(\)](#)  
[object\\_attach\\_byptr\(\)](#)

### 26.39.3.10 `t_class* object_class (void * x)`

Determines the class of a given object.

**Parameters:**

*x* The object to test

**Returns:**

This function returns the [t\\_class](#) \* of the object's class, if successful, or NULL, if unsuccessful.

**26.39.3.11 t\_symbol\* object\_classname (void \* x)**

Retrieves an object instance's class name.

**Parameters:**

*x* The object instance whose class name is being queried

**Returns:**

The classname, or NULL if unsuccessful.

**26.39.3.12 long object\_classname\_compare (void \* x, t\_symbol \* name)**

Determines if a particular object is an instance of a given class.

**Parameters:**

*x* The object to test

*name* 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"));
```

**26.39.3.13 t\_max\_err object\_detach (t\_symbol \* name\_space, t\_symbol \* s, void \* x)**

Detach a client from a registered object.

**Parameters:**

*name\_space* The namespace of the registered object. This should be the same value used in [object\\_register\(\)](#) to register the object. If you don't know the registered object's namespace, the [object\\_findregisteredbyptr\(\)](#) function can be used to determine it.

*s* The name of the registered object in the namespace. If you don't know the name of the registered object, the [object\\_findregisteredbyptr\(\)](#) function can be used to determine it.

*x* 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.

**26.39.3.14 t\_max\_err object\_detach\_byptr (void \* *x*, void \* *registeredobject*)**

Detach a client from a registered object.

**Parameters:**

*x* The attaching client object. Generally, this is the pointer to your Max object.

*registeredobject* The object from which to detach.

**Returns:**

A Max error code.

**See also:**

[object\\_detach\(\)](#)

[object\\_attach\\_byptr\(\)](#)

**26.39.3.15 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:**

*ac* The number of atoms.

*av* A pointer to the first atom in the array.

**Returns:**

The dictionary retrieved from the atoms.

**See also:**

[attr\\_dictionary\\_process\(\)](#)

**26.39.3.16 void\* object\_findregistered (t\_symbol \* *name\_space*, t\_symbol \* *s*)**

Determines a registered object's pointer, given its namespace and name.

**Parameters:**

*name\_space* The namespace of the registered object

*s* 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.

### 26.39.3.17 `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:

- name\_space* Pointer to a `t_symbol *`, to receive the namespace of the registered object
- s* Pointer to a `t_symbol *`, to receive the name of the registered object within the namespace
- x* 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.

### 26.39.3.18 `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:

- x* 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 `db_close()`, and `db_open()`.

### 26.39.3.19 `method object_getmethod (void * x, t_symbol * s)`

Retrieves an object's `method` for a particular message selector.

#### Parameters:

- x* The object whose method is being queried
- s* The message selector

#### Returns:

This function returns the `method` if successful, or 0 if unsuccessful.

### 26.39.3.20 `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:**

- x* The object whose value is of interest
- ac* Pointer to a long variable to receive the count of arguments in *av*. The long variable itself should be set to 0 previous to calling this function.
- av* Pointer to a `t_atom *`, to receive object data. The `t_atom *` 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_addmethod(c, (method)myobject_getvalueof, "getvalueof", A_CANT, 0);
```

The `getvalueof` method should be prototyped as:

```
t_max_err myobject_getvalueof(t_myobject *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.
```

**26.39.3.21 void\* object\_method (void \*x, t\_symbol \*s, ...)**

Sends an untyped message to an object.

**Parameters:**

- x* The object that will receive the message
- s* 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 `db_query()`, `db_query_getlastinsertid()`, `db_result_clear()`, `db_result_fieldname()`, `db_result_float()`, `db_result_long()`, `db_result_nextrecord()`, `db_result_numfields()`, `db_result_numrecords()`, `db_result_reset()`, `db_result_string()`, `db_transaction_end()`, `db_transaction_flush()`, `db_transaction_start()`, `db_view_create()`, `db_view_getresult()`, and `db_view_remove()`.

### 26.39.3.22 `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:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- v* An argument to pass to the method.
- rv* The address of an atom to hold a return value.

**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)

### 26.39.3.23 `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:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- ac* The number of arguments to pass to the method.
- av* The address of the first of the array of arguments to pass to the method.
- rv* The address of an atom to hold a return value.

**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)

**26.39.3.24 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:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- v* An argument to pass to the method.
- rv* The address of an atom to hold a return value.

**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)

**26.39.3.25 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:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- ac* The number of arguments to pass to the method.
- av* The address of the first of the array of arguments to pass to the method.
- rv* The address of an atom to hold a return value.

**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)

**26.39.3.26 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:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- v* An argument to pass to the method.
- rv* The address of an atom to hold a return value.



**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)

**26.39.3.27 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:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- ac* The number of arguments to pass to the method.
- av* The address of the first of the array of arguments to pass to the method.
- rv* The address of an atom to hold a return value.

**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)

**26.39.3.28 t\_max\_err object\_method\_format (t\_object \* *x*, t\_symbol \* *s*, t\_atom \* *rv*, char \* *fmt*, ...)**

Convenience wrapper for [object\\_method\\_typed\(\)](#) that uses [atom\\_setformat\(\)](#) to define the arguments.

**Parameters:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- rv* The address of an atom to hold a return value.
- fmt* 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\(\)](#)

**26.39.3.29 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:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- v* An argument to pass to the method.
- rv* The address of an atom to hold a return value.

**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)

**26.39.3.30 t\_max\_err object\_method\_long\_array (t\_object \* *x*, t\_symbol \* *s*, long *ac*, long \* *av*, t\_atom \* *rv*)**

Convenience wrapper for [object\\_method\\_typed\(\)](#) that passes an array of long integers values as an argument.

**Parameters:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- ac* The number of arguments to pass to the method.
- av* The address of the first of the array of arguments to pass to the method.
- rv* The address of an atom to hold a return value.

**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)

**26.39.3.31 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:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- v* An argument to pass to the method.
- rv* The address of an atom to hold a return value.

**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)

**26.39.3.32** `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:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- ac* The number of arguments to pass to the method.
- av* The address of the first of the array of arguments to pass to the method.
- rv* The address of an atom to hold a return value.

**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)

**26.39.3.33** `t_max_err object_method_parse (t_object * x, t_symbol * s, char * parsestr, t_atom *  
rv)`

Convenience wrapper for [object\\_method\\_typed\(\)](#) that uses [atom\\_setparse\(\)](#) to define the arguments.

**Parameters:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- parsestr* A C-string to parse into an array of atoms to pass to the method.
- rv* The address of an atom to hold a return value.

**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)  
[atom\\_setparse\(\)](#)

**26.39.3.34 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:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- v* An argument to pass to the method.
- rv* The address of an atom to hold a return value.

**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)

**26.39.3.35 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:**

- x* The object to which the message will be sent.
- s* The name of the method to call on the object.
- ac* The number of arguments to pass to the method.
- av* The address of the first of the array of arguments to pass to the method.
- rv* The address of an atom to hold a return value.

**Returns:**

A Max error code.

**See also:**

[object\\_method\\_typed\(\)](#)

**26.39.3.36 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:**

- x* The object that will receive the message
- s* The message selector
- ac* Count of message arguments in *av*
- av* Array of *t\_atoms*; the message arguments

*rv* 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.

#### 26.39.3.37 [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:

*x* The object that will receive the message  
*mp* Undocumented  
*s* The message selector  
*ac* Count of message arguments in *av*  
*av* Array of [t\\_atoms](#); the message arguments  
*rv* 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.

#### 26.39.3.38 [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:

*name\_space* The desired object's name space. Typically, either the constant [CLASS\\_BOX](#), for obex classes which can instantiate inside of a Max patcher (e.g. boxes, UI objects, etc.), or the constant [CLASS\\_NOBOX](#), for classes which will only be used internally. Developers can define their own name spaces as well, but this functionality is currently undocumented.  
*classname* 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.

### 26.39.3.39 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:

***name\_space*** The desired object's name space. Typically, either the constant [CLASS\\_BOX](#), for obex classes which can instantiate inside of a Max patcher (e.g. boxes, UI objects, etc.), or the constant [CLASS\\_NOBOX](#), for classes which will only be used internally. Developers can define their own name spaces as well, but this functionality is currently undocumented.

***classname*** The name of the class of the object to be created

***ac*** Count of arguments in *av*

***av*** Array of t\_atoms; 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.

Referenced by [db\\_open\(\)](#).

### 26.39.3.40 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:

***x*** Pointer to the registered object

***s*** The message to send

***data*** 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.

**26.39.3.41 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:**

- x* The object pointer. This function should only be called on instantiated objects (i.e. in the `new` method or later), not directly on classes (i.e. in `main()`).
- s* The message selector [t\\_symbol \\*](#)
- argc* Number of elements in the argument list in `argv`
- argv* `t_atoms` 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.

**26.39.3.42 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:**

- x* The object pointer. This function should only be called on instantiated objects (i.e. in the `new` method or later), not directly on classes (i.e. in `main()`).
- key* The symbolic name for the data to be retrieved
- val* A pointer to a [t\\_object \\*](#), 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;

    p = object_obex_lookup(x, gensym("#P"), (t_object **)&p);
    b = 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);
}
```

**26.39.3.43 t\_max\_err object\_obex\_store (void \*x, t\_symbol \*key, t\_object \*val)**

Stores data in the object's obex.

**Parameters:**

*x* The object pointer. This function should only be called on instantiated objects (i.e. in the `new` method or later), not directly on classes (i.e. in `main()`).

*key* A symbolic name for the data to be stored

*val* A `t_object *`, to be stored in the obex, referenced under the *key*.

**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));
```

**26.39.3.44 void object\_openhelp (t\_object \* x)**

Open the help patcher for a given instance of an object.

**Parameters:**

*x* The object instance for which to open the help patcher.

**26.39.3.45 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.

**26.39.3.46 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.

**26.39.3.47 void\* object\_register (t\_symbol \* name\_space, t\_symbol \* s, void \* x)**

Registers an object in a namespace.

**Parameters:**

*name\_space* 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.



*s* The name of the object in the namespace. This name will be used by other objects to attach and detach from the registered object.

*x* 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 `bjct_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()`.

### 26.39.3.48 `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:

*x* The object whose value is of interest

*ac* The count of arguments in *av*

*av* Array of `t_atoms`; 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_addmethod(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;
}
```

### 26.39.3.49 `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 [object\\_register\(\)](#) function.

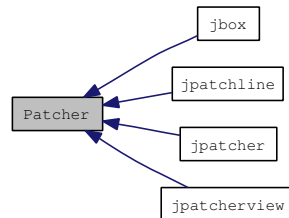
**Returns:**

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.

## 26.40 Patcher

Max's patcher represents a graph of objects that communicate with messages.

Collaboration diagram for Patcher:



### Data Structures

- struct [t\\_jbox](#)

*The [t\\_jbox](#) struct provides the header for a Max user-interface object.*

### Modules

- [jpatcher](#)

*The patcher.*

- [jbox](#)

*A box in the patcher.*

- [jpatchline](#)

*A patch cord.*

- [jpatcherview](#)

*A view of a patcher.*

### Typedefs

- typedef [t\\_object](#) [t\\_patcher](#)

*A patcher.*

- typedef [t\\_object](#) [t\\_box](#)

*A box.*

### Enumerations

- enum {  
    [PI\\_DEEP](#) = 1,

```

PI_REQUIREFIRSTIN = 2,
PI_WANTBOX = 4 }
    patcher iteration flags

```

### 26.40.1 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.

### 26.40.2 Typedef Documentation

#### 26.40.2.1 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.

Definition at line 30 of file `ext_maxtypes.h`.

#### 26.40.2.2 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.

Definition at line 23 of file `ext_maxtypes.h`.

### 26.40.3 Enumeration Type Documentation

#### 26.40.3.1 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

```

Definition at line 48 of file `ext_maxtypes.h`.

## 26.41 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 \(t\\_object \\*p, char c\)](#)  
*Set whether a patcher's foreground layer is hidden.*
- [t\\_max\\_err jpatcher\\_get\\_editing\\_bgcolor \(t\\_object \\*p, t\\_jrgba \\*prgba\)](#)  
*Retrieve a patcher's editing background color.*
- [t\\_max\\_err jpatcher\\_set\\_editing\\_bgcolor \(t\\_object \\*p, t\\_jrgba \\*prgba\)](#)  
*Set a patcher's editing background color.*
- [t\\_max\\_err jpatcher\\_get\\_bgcolor \(t\\_object \\*p, t\\_jrgba \\*prgba\)](#)  
*Retrieve a patcher's locked background color.*
- [t\\_max\\_err jpatcher\\_set\\_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_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.*

### 26.41.1 Detailed Description

The patcher.

### 26.41.2 Function Documentation

#### 26.41.2.1 `void jpatcher_deleteobj (t_object *p, t_jbox *b)`

Delete an object that is in a patcher.

**Parameters:**

- p* The patcher.
- b* The object box to delete.

**26.41.2.2 t\_max\_err jpatcher\_get\_bgcolor (t\_object \* *p*, t\_jrgba \* *prgba*)**

Retrieve a patcher's locked background color.

**Parameters:**

- p* The patcher to be queried.
- prgba* The address of a valid [t\\_jrgba](#) struct that will be filled-in with the current patcher color values.

**Returns:**

A Max error code.

**26.41.2.3 char jpatcher\_get\_bghidden (t\_object \* *p*)**

Determine whether a patcher's background layer is hidden.

**Parameters:**

- p* The patcher to be queried.

**Returns:**

True if the background layer is hidden, otherwise false.

**26.41.2.4 char jpatcher\_get\_bglocked (t\_object \* *p*)**

Determine whether a patcher's background layer is locked.

**Parameters:**

- p* The patcher to be queried.

**Returns:**

True if the background layer is locked, otherwise false.

**26.41.2.5 t\_object\* jpatcher\_get\_box (t\_object \* *p*)**

If a patcher is inside a box, return its box.

**Parameters:**

- p* The patcher to be queried.

**Returns:**

A pointer to the box containing the patcher, otherwise NULL.



**26.41.2.6 long jpatcher\_get\_count (t\_object \* p)**

Determine the number of boxes in a patcher.

**Parameters:**

*p* The patcher to be queried.

**Returns:**

The number of boxes in the patcher.

**26.41.2.7 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.

**26.41.2.8 long jpatcher\_get\_default\_fontface (t\_object \* p)**

Return the index of the default font face used for new objects in a patcher.

**Parameters:**

*p* A pointer to a patcher instance.

**Returns:**

The index of the default font face used for new objects in a patcher.

**26.41.2.9 t\_symbol\* jpatcher\_get\_default\_fontname (t\_object \* p)**

Return the name of the default font used for new objects in a patcher.

**Parameters:**

*p* A pointer to a patcher instance.

**Returns:**

The name of the default font used for new objects in a patcher.

**26.41.2.10 float jpatcher\_get\_default\_fontsize (t\_object \* p)**

Return the size of the default font used for new objects in a patcher.

**Parameters:**

*p* A pointer to a patcher instance.

**Returns:**

The size of the default font used for new objects in a patcher.

**26.41.2.11 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:**

- p* A pointer to a patcher instance.
- pr* The address of valid [t\\_rect](#) whose values will be filled-in upon return.

**Returns:**

A Max error code.

**26.41.2.12 char jpatcher\_get\_dirty (t\_object \* *p*)**

Determine whether a patcher's dirty bit has been set.

**Parameters:**

- p* The patcher to be queried.

**Returns:**

True if the patcher is dirty, otherwise false.

**26.41.2.13 t\_max\_err jpatcher\_get\_editing\_bgcolor (t\_object \* *p*, t\_jrgba \* *prgba*)**

Retrieve a patcher's editing background color.

**Parameters:**

- p* The patcher to be queried.
- prgba* The address of a valid [t\\_jrgba](#) struct that will be filled-in with the current patcher color values.

**Returns:**

A Max error code.

**26.41.2.14 char jpatcher\_get\_fghidden (t\_object \* *p*)**

Determine whether a patcher's foreground layer is hidden.

**Parameters:**

- p* The patcher to be queried.

**Returns:**

True if the foreground layer is hidden, otherwise false.

**26.41.2.15 t\_symbol\* jpatcher\_get\_filename (t\_object \* p)**

Retrieve a patcher's file name.

**Parameters:**

*p* The patcher to be queried.

**Returns:**

The patcher's file name.

**26.41.2.16 t\_symbol\* jpatcher\_get\_filepath (t\_object \* p)**

Retrieve a patcher's file path.

**Parameters:**

*p* The patcher to be queried.

**Returns:**

The patcher's file path.

**26.41.2.17 long jpatcher\_get\_fileversion (t\_object \* p)**

Return the file version of the patcher.

**Parameters:**

*p* A pointer to the patcher whose version number is desired.

**Returns:**

The file version number.

**26.41.2.18 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:**

*p* The patcher to be queried.

**Returns:**

The first jpatchline in a patcher.

**26.41.2.19 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:**

*p* The patcher to be queried.

**Returns:**

The first box in a patcher.

**See also:**

[jbox\\_get\\_prevobject\(\)](#) [jbox\\_get\\_nextobject\(\)](#) [jpatcher\\_get\\_lastobject\(\)](#)

**26.41.2.20 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:**

*p* The patcher to be queried.

**Returns:**

The first view of a patcher.

**26.41.2.21 t\_max\_err jpatcher\_get\_gridsize (t\_object \* p, double \* gridsizeX, double \* gridsizeY)**

Retrieve a patcher's grid size.

**Parameters:**

*p* The patcher to be queried.

*gridsizeX* The address of a double that will be set to the current horizontal grid spacing for the patcher.

*gridsizeY* The address of a double that will be set to the current vertical grid spacing for the patcher.

**Returns:**

A Max error code.

**26.41.2.22 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:**

*p* The patcher to be queried.

**Returns:**

The last box in a patcher.

**See also:**

[jbox\\_get\\_prevobject\(\)](#) [jbox\\_get\\_nextobject\(\)](#) [jpatcher\\_get\\_firstobject\(\)](#)

**26.41.2.23 t\_symbol\* jpatcher\_get\_name (t\_object \* p)**

Retrieve a patcher's name.

**Parameters:**

*p* The patcher to be queried.

**Returns:**

The patcher's name.

**26.41.2.24 t\_object\* jpatcher\_get\_parentpatcher (t\_object \* p)**

Given a patcher, return its parent patcher.

**Parameters:**

*p* 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.

**26.41.2.25 char jpatcher\_get\_presentation (t\_object \* p)**

Determine whether a patcher is currently in presentation mode.

**Parameters:**

*p* The patcher to be queried.

**Returns:**

True if the patcher is in presentation mode, otherwise false.

**26.41.2.26 t\_max\_err jpatcher\_get\_rect (t\_object \* p, t\_rect \* pr)**

Query a patcher to determine its location and size.

**Parameters:**

*p* A pointer to a patcher instance.

*pr* The address of valid [t\\_rect](#) whose values will be filled-in upon return.

**Returns:**

A Max error code.

**26.41.2.27 t\_symbol\* jpatcher\_get\_title (t\_object \* *p*)**

Retrieve a patcher's title.

**Parameters:**

*p* The patcher to be queried.

**Returns:**

The patcher's title.

**26.41.2.28 t\_object\* jpatcher\_get\_toppatcher (t\_object \* *p*)**

Given a patcher, return the top-level patcher for the tree in which it exists.

**Parameters:**

*p* The patcher to be queried.

**Returns:**

The patcher's top-level parent patcher.

**26.41.2.29 int jpatcher\_is\_patcher (t\_object \* *p*)**

Determine if a [t\\_object\\*](#) is a patcher object.

**Parameters:**

*p* The object pointer to test.

**Returns:**

Returns true if the object is a patcher, otherwise returns non-zero.

**26.41.2.30 t\_max\_err jpatcher\_set\_bgcolor (t\_object \* *p*, t\_jrgba \* *prgba*)**

Set a patcher's locked background color.

**Parameters:**

*p* The patcher to be queried.

*prgba* The address of a [t\\_jrgba](#) struct containing the new color to use.

**Returns:**

A Max error code.

**26.41.2.31 t\_max\_err jpatcher\_set\_bghidden (t\_object \* *p*, char *c*)**

Set whether a patcher's background layer is hidden.

**Parameters:**

- p* The patcher whose dirty bit will be set.
- c* Pass true to hide the patcher's background layer, otherwise pass false.

**Returns:**

A Max error code.

**26.41.2.32 t\_max\_err jpatcher\_set\_bglocked (t\_object \* *p*, char *c*)**

Set whether a patcher's background layer is locked.

**Parameters:**

- p* The patcher whose dirty bit will be set.
- c* Pass true to lock the patcher's background layer, otherwise pass false.

**Returns:**

A Max error code.

**26.41.2.33 t\_max\_err jpatcher\_set\_defrect (t\_object \* *p*, t\_rect \* *pr*)**

Set a patcher's default location and size.

**Parameters:**

- p* A pointer to a patcher instance.
- pr* The address of a [t\\_rect](#) with the new position and size.

**Returns:**

A Max error code.

**26.41.2.34 t\_max\_err jpatcher\_set\_dirty (t\_object \* *p*, char *c*)**

Set a patcher's dirty bit.

**Parameters:**

- p* The patcher whose dirty bit will be set.
- c* The new value for the patcher's dirty bit (pass true or false).

**Returns:**

A Max error code.

**26.41.2.35 t\_max\_err jpatcher\_set\_editing\_bgcolor (t\_object \* *p*, t\_jrgba \* *prgba*)**

Set a patcher's editing background color.

**Parameters:**

*p* The patcher to be queried.

*prgba* The address of a [t\\_jrgba](#) struct containing the new color to use.

**Returns:**

A Max error code.

**26.41.2.36 t\_max\_err jpatcher\_set\_fghidden (t\_object \* *p*, char *c*)**

Set whether a patcher's foreground layer is hidden.

**Parameters:**

*p* The patcher whose dirty bit will be set.

*c* Pass true to hide the patcher's foreground layer, otherwise pass false.

**Returns:**

A Max error code.

**26.41.2.37 t\_max\_err jpatcher\_set\_gridsize (t\_object \* *p*, double *gridsizeX*, double *gridsizeY*)**

Set a patcher's grid size.

**Parameters:**

*p* The patcher to be queried.

*gridsizeX* The new horizontal grid spacing for the patcher.

*gridsizeY* The new vertical grid spacing for the patcher.

**Returns:**

A Max error code.

**26.41.2.38 t\_max\_err jpatcher\_set\_locked (t\_object \* *p*, char *c*)**

Lock or unlock a patcher.

**Parameters:**

*p* The patcher whose locked state will be changed.

*c* Pass true to lock a patcher, otherwise pass false.

**Returns:**

A Max error code.



**26.41.2.39 t\_max\_err jpatcher\_set\_presentation (t\_object \* *p*, char *c*)**

Set a patcher to presentation mode.

**Parameters:**

- p* The patcher whose locked state will be changed.
- c* Pass true to switch the patcher to presentation mode, otherwise pass false.

**Returns:**

A Max error code.

**26.41.2.40 t\_max\_err jpatcher\_set\_rect (t\_object \* *p*, t\_rect \* *pr*)**

Set a patcher's location and size.

**Parameters:**

- p* A pointer to a patcher instance.
- pr* The address of a [t\\_rect](#) with the new position and size.

**Returns:**

A Max error code.

**26.41.2.41 t\_max\_err jpatcher\_set\_title (t\_object \* *p*, t\_symbol \* *ps*)**

Set a patcher's title.

**Parameters:**

- p* The patcher whose locked state will be changed.
- ps* The new title for the patcher.

**Returns:**

A Max error code.

**26.41.2.42 t\_symbol\* jpatcher\_uniqueboxname (t\_object \* *p*, t\_symbol \* *classname*)**

Generate a unique name for a box in patcher.

**Parameters:**

- p* A pointer to a patcher instance.
- classname* 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.

## 26.42 jbox

A box in the patcher.

Collaboration diagram for jbox:



### Data Structures

- struct [t\\_jboxdrawparams](#)  
The [t\\_jboxdrawparams](#) structure.

### Defines

- #define [JBOX\\_DRAWFIRSTIN](#) (1<<0)  
*draw first inlet*
- #define [JBOX\\_NODRAWBOX](#) (1<<1)  
*don't draw the frame*
- #define [JBOX\\_DRAWINLAST](#) (1<<2)  
*draw inlets after update method*
- #define [JBOX\\_TRANSPARENT](#) (1<<3)  
*don't make transparent unless you need it (for efficiency)*
- #define [JBOX\\_NOGROW](#) (1<<4)  
*don't even draw grow thingie*
- #define [JBOX\\_GROWY](#) (1<<5)  
*can grow in y direction by dragging*
- #define [JBOX\\_GROWBOTH](#) (1<<6)  
*can grow independently in both x and y*
- #define [JBOX\\_IGNORELOCKCLICK](#) (1<<7)  
*box should ignore a click if patcher is locked*
- #define [JBOX\\_HILITE](#) (1<<8)  
*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](#) (1<<9)  
*immediately set box into the background*
- #define [JBOX\\_NOFLOATINSPECTOR](#) (1<<10)

*no floating inspector window*

- #define **JBOX\_TEXTFIELD** (1<<11)  
*save/load text from textfield, unless JBOX\_BINBUF flag is set*
- #define **JBOX\_FIXWIDTH** (1<<19)  
*give the box a textfield based fix-width (bfixwidth) method*
- #define **JBOX\_FONTATTR** (1<<18)  
*if you want font related attribute you must add this to jbox\_initclass()*
- #define **JBOX\_BINBUF** (1<<14)  
*save/load text from b\_binbuf*
- #define **JBOX\_MOUSEDRAGDELTA** (1<<12)  
*hides mouse cursor in drag and sends mousedragdelta instead of mousedrag (for infinite scrolling like number)*
- #define **JBOX\_COLOR** (1<<13)  
*support the "color" method for color customization*
- #define **JBOX\_DRAWIOLOCKED** (1<<15)  
*draw inlets and outlets when locked (default is not to draw them)*
- #define **JBOX\_DRAWBACKGROUND** (1<<16)  
*set to have box bg filled in for you based on getdrawparams method or brgba attribute*
- #define **JBOX\_NOINSPECTFIRSTIN** (1<<17)  
*flag for objects such as bpatcher that have a different b\_firstin,*
- #define **JBOX\_DEFAULTNAMES** (1<<18)  
*flag instructing jbox\_new to attach object to the defaults object for live defaults updating*
- #define **JBOX\_FOCUS** (1<<20)  
*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*

## Enumerations

- enum {  
**JBOX\_FONTFACE\_REGULAR** = 0,  
**JBOX\_FONTFACE\_BOLD** = 1,  
**JBOX\_FONTFACE\_ITALIC** = 2,  
**JBOX\_FONTFACE\_BOLDITALIC** = 3 }  
*actual numerical values of the b\_fontface attribute; use jbox\_fontface() to weight*

- enum `HitTestResult` {  
`HitNothing` = 0,  
`HitBox` = 1,  
`HitInlet` = 2,  
`HitOutlet` = 3,  
`HitGrowBox` = 4,  
`HitLine` = 5 }

*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)`  
*Set the position of a box for both the presentation and patching views.*
- `t_max_err jbox_get_patching_position (t_object *box, t_pt *pos)`  
*Fetch the position of a box for the patching view.*
- `t_max_err jbox_set_patching_position (t_object *box, t_pt *pos)`  
*Set the position of a box for the patching view.*

- `t_max_err jbox_get_presentation_position (t_object *box, t_pt *pos)`  
*Fetch the position of a box for the presentation view.*
- `t_max_err jbox_set_presentation_position (t_object *box, t_pt *pos)`  
*Set 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_jbox *b, long flags, long argc, t_atom *argv)`  
*Set up your UI object's `t_jbox` member.*
- `void jbox_free (t_jbox *b)`  
*Tear down your UI object's `t_jbox` member.*
- `void jbox_ready (t_jbox *b)`  
*Mark the box ready to be accessed and drawn by Max.*
- `void jbox_redraw (t_jbox *b)`  
*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.*

### 26.42.1 Detailed Description

A box in the patcher.

## 26.42.2 Define Documentation

### 26.42.2.1 #define JBOX\_NOINSPECTFIRSTIN (1<<17)

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

Definition at line 1646 of file jpatcher\_api.h.

## 26.42.3 Enumeration Type Documentation

### 26.42.3.1 anonymous enum

actual numerical values of the b\_fontface attribute; use jbox\_fontface() to weight

Enumerator:

*JBOX\_FONTFACE\_REGULAR* normal  
*JBOX\_FONTFACE\_BOLD* bold  
*JBOX\_FONTFACE\_ITALIC* italic  
*JBOX\_FONTFACE\_BOLDITALIC* bold and italic

Definition at line 1657 of file jpatcher\_api.h.

### 26.42.3.2 enum HitTestResult

enumerations used for box decorators

Enumerator:

*HitNothing* a hole in the box  
*HitBox* the body of the box  
*HitInlet* an inlet  
*HitOutlet* an outlet  
*HitGrowBox* the grow handle  
*HitLine* a line

Definition at line 1670 of file jpatcher\_api.h.

## 26.42.4 Function Documentation

### 26.42.4.1 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:

*b* The address of your object's t\_jbox member (which should be the first member of the object's struct).



**26.42.4.2 char\* jbox\_get\_annotation (t\_object \* *bb*)**

Retrieve a box's annotation string, if the user has given it an annotation.

**Parameters:**

*bb* The box to query.

**Returns:**

The user-created annotation string for a box, or NULL if no string exists.

**26.42.4.3 char jbox\_get\_background (t\_object \* *b*)**

Determine whether a box is located in the patcher's background layer.

**Parameters:**

*b* The box to query.

**Returns:**

Zero if the object is in the foreground, otherwise non-zero.

**26.42.4.4 char jbox\_get\_canhilite (t\_object \* *b*)**

Retrieve a box flag value from a box.

**Parameters:**

*b* The box to query.

**Returns:**

The value of the canhilite bit in the box's flags.

**26.42.4.5 t\_max\_err jbox\_get\_color (t\_object \* *b*, t\_jrgba \* *prgba*)**

Retrieve a box's 'color' attribute.

**Parameters:**

*b* The box to query.

*prgba* The address of a valid [t\\_rect](#) whose values will be filled-in upon return.

**Returns:**

A Max error code.

**26.42.4.6 char jbox\_get\_drawfirstin (t\_object \* *b*)**

Determine whether a box draws its first inlet.

**Parameters:**

*b* The box to query.

**Returns:**

Zero if the inlet is not drawn, otherwise non-zero.

**26.42.4.7 char jbox\_get\_drawinlast (t\_object \* *b*)**

Retrieve a box flag value from a box.

**Parameters:**

*b* The box to query.

**Returns:**

The value of the drawinlast bit in the box's flags.

**26.42.4.8 t\_symbol\* jbox\_get\_fontname (t\_object \* *b*)**

Retrieve a box's 'fontname' attribute.

**Parameters:**

*b* The box to query.

**Returns:**

The font name.

**26.42.4.9 double jbox\_get\_fontsize (t\_object \* *b*)**

Retrieve a box's 'fontsize' attribute.

**Parameters:**

*b* The box to query.

**Returns:**

The font size in points.

**26.42.4.10 char jbox\_get\_growboth (t\_object \* *b*)**

Retrieve a box flag value from a box.

**Parameters:**

*b* The box to query.

**Returns:**

The value of the growboth bit in the box's flags.

**26.42.4.11 char jbox\_get\_growy (t\_object \* *b*)**

Retrieve a box flag value from a box.

**Parameters:**

*b* The box to query.

**Returns:**

The value of the growy bit in the box's flags.

**26.42.4.12 char jbox\_get\_hidden (t\_object \* *b*)**

Retrieve a box's 'hidden' attribute.

**Parameters:**

*b* The box to query.

**Returns:**

True if the box is hidden, otherwise false.

**26.42.4.13 t\_symbol\* jbox\_get\_hint (t\_object \* *b*)**

Retrieve a box's hint text as a symbol.

**Parameters:**

*b* The box to query.

**Returns:**

The box's hint text.

**26.42.4.14** `char* jbox_get_hintstring (t_object * bb)`

Retrieve a box's hint text as a C-string.

**Parameters:**

*bb* The box to query.

**Returns:**

The box's hint text.

**26.42.4.15** `t_symbol* jbox_get_id (t_object * b)`

Retrieve a boxes unique id.

**Parameters:**

*b* The box to query.

**Returns:**

The unique id of the object. This is a symbol that is referenced, for example, by patchlines.

**26.42.4.16** `char jbox_get_ignoreclick (t_object * b)`

Determine whether a box ignores clicks.

**Parameters:**

*b* The box to query.

**Returns:**

Zero if the object responds to clicks, otherwise non-zero.

**26.42.4.17** `t_symbol* jbox_get_maxclass (t_object * b)`

Retrieve the name of the class of the box's object.

**Parameters:**

*b* The box to query.

**Returns:**

The name of the class of the box's object.

**26.42.4.18 t\_object\* jbox\_get\_nextobject (t\_object \* b)**

The next box in the patcher's (linked) list of boxes.

**Parameters:**

*b* The box to query.

**Returns:**

The next box in the list.

**26.42.4.19 char jbox\_get\_nogrow (t\_object \* b)**

Retrieve a box flag value from a box.

**Parameters:**

*b* The box to query.

**Returns:**

The value of the nogrow bit in the box's flags.

**26.42.4.20 t\_object\* jbox\_get\_object (t\_object \* b)**

Retrieve a pointer to the box's object.

**Parameters:**

*b* The box to query.

**Returns:**

A pointer to the box's object.

**26.42.4.21 char jbox\_get\_outline (t\_object \* b)**

Determine whether a box draws an outline.

**Parameters:**

*b* The box to query.

**Returns:**

Zero if the outline is not drawn, otherwise non-zero.

**26.42.4.22 t\_object\* jbox\_get\_patcher (t\_object \* *b*)**

Retrieve a box's patcher.

**Parameters:**

*b* The box to query.

**Returns:**

If the box has a patcher, the patcher's pointer is returned. Otherwise NULL is returned.

**26.42.4.23 t\_max\_err jbox\_get\_patching\_position (t\_object \* *box*, t\_pt \* *pos*)**

Fetch the position of a box for the patching view.

**Parameters:**

*box* The box whose position will be retrieved.

*pos* The address of a valid [t\\_pt](#) whose x and y values will be filled in.

**Returns:**

A Max error code.

**26.42.4.24 t\_max\_err jbox\_get\_patching\_rect (t\_object \* *box*, t\_rect \* *pr*)**

Retrieve the patching rect of a box.

**Parameters:**

*box* The box whose rect values will be retrieved.

*pr* The address of a valid [t\\_rect](#) whose values will be filled in.

**Returns:**

A Max error code.

**26.42.4.25 t\_max\_err jbox\_get\_patching\_size (t\_object \* *box*, t\_size \* *size*)**

Fetch the size of a box for the patching view.

**Parameters:**

*box* The box whose size will be retrieved.

*size* The address of a valid [t\\_size](#) whose width and height values will be filled in.

**Returns:**

A Max error code.

**26.42.4.26 char jbox\_get\_presentation (t\_object \* *b*)**

Determine if a box is included in the presentation view.

**Parameters:**

*b* The box to query.

**Returns:**

Non-zero if in presentation mode, otherwise zero.

**26.42.4.27 t\_max\_err jbox\_get\_presentation\_position (t\_object \* *box*, t\_pt \* *pos*)**

Fetch the position of a box for the presentation view.

**Parameters:**

*box* The box whose position will be retrieved.

*pos* The address of a valid [t\\_pt](#) whose x and y values will be filled in.

**Returns:**

A Max error code.

**26.42.4.28 t\_max\_err jbox\_get\_presentation\_rect (t\_object \* *box*, t\_rect \* *pr*)**

Retrieve the presentation rect of a box.

**Parameters:**

*box* The box whose rect values will be retrieved.

*pr* The address of a valid [t\\_rect](#) whose values will be filled in.

**Returns:**

A Max error code.

**26.42.4.29 t\_max\_err jbox\_get\_presentation\_size (t\_object \* *box*, t\_size \* *size*)**

Fetch the size of a box for the presentation view.

**Parameters:**

*box* The box whose size will be retrieved.

*size* The address of a valid [t\\_size](#) whose width and height values will be filled in.

**Returns:**

A Max error code.

**26.42.4.30 t\_object\* jbox\_get\_prevobject (t\_object \* b)**

The previous box in the patcher's (linked) list of boxes.

**Parameters:**

*b* The box to query.

**Returns:**

The next box in the list.

**26.42.4.31 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:**

*box* The box whose rect will be fetched.

*which* The name of the rect attribute to be fetched, for example `_sym_presentation_rect` or `_sym_patching_rect`.

*pr* The address of a valid `t_rect` whose members will be filled in by this function.

**Returns:**

A Max error code.

**26.42.4.32 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:**

*box* The box whose rect will be fetched.

*patcherview* A patcherview in which the box exists.

*rect* The address of a valid `t_rect` whose members will be filled in by this function.

**Returns:**

A Max error code.

**26.42.4.33 t\_object\* jbox\_get\_textfield (t\_object \* b)**

Retrieve a pointer to a box's textfield.

**Parameters:**

*b* 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.



**26.42.4.34 t\_symbol\* jbox\_get\_varname (t\_object \* b)**

Retrieve a box's scripting name.

**Parameters:**

*b* The box to query.

**Returns:**

The box's scripting name.

**26.42.4.35 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:**

*b* The address of your UI object's [t\\_jbox](#) member (which should be the first member of the object's struct).

*flags* Flags to set the box's behavior, such as [JBOX\\_NODRAWBOX](#).

*argc* The count of atoms in the argv parameter.

*argv* 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.

**26.42.4.36 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:**

*b* The address of your object's [t\\_jbox](#) member.

*s* The name of the send object.

*msg* The notification name.

*sender* The sending object's address.

*data* A pointer to some data passed to the box's notify method.

**Returns:**

A Max error code.

**26.42.4.37 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:**

*b* The address of your object's [t\\_jbox](#) member.

**26.42.4.38 void jbox\_redraw (t\_jbox \* b)**

Request that your object/box be re-drawn by Max.

**Parameters:**

*b* The address of your object's [t\\_jbox](#) member.

**26.42.4.39 void jbox\_set\_annotation (t\_object \* bb, char \* s)**

Set a box's annotation string.

**Parameters:**

*bb* The box to query.

*s* The annotation string for the box.

**Returns:**

A Max error code.

**26.42.4.40 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:**

*b* The box to query.

*c* 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.

**26.42.4.41 t\_max\_err jbox\_set\_color (t\_object \* b, t\_jrgba \* prgba)**

Set a box's 'color' attribute.

**Parameters:**

*b* The box to query.

*prgba* The address of a [t\\_rect](#) containing the desired color for the box/object.

**Returns:**

A Max error code.

**26.42.4.42 t\_max\_err jbox\_set\_fontname (t\_object \* *b*, t\_symbol \* *ps*)**

Set a box's 'fontname' attribute.

**Parameters:**

*b* The box to query.

*ps* The font name. Note that the font name may be case-sensitive.

**Returns:**

A Max error code.

**26.42.4.43 t\_max\_err jbox\_set\_fontsize (t\_object \* *b*, double *d*)**

Set a box's 'fontsize' attribute.

**Parameters:**

*b* The box to query.

*d* The fontsize in points.

**Returns:**

A Max error code.

**26.42.4.44 t\_max\_err jbox\_set\_hidden (t\_object \* *b*, char *c*)**

Set a box's 'hidden' attribute.

**Parameters:**

*b* The box to query.

*c* Set to true to hide the box, otherwise false.

**Returns:**

A Max error code.

**26.42.4.45 t\_max\_err jbox\_set\_hint (t\_object \* *b*, t\_symbol \* *s*)**

Set a box's hint text using a symbol.

**Parameters:**

*b* The box to query.

*s* The new text to use for the box's hint.

**Returns:**

A Max error code.

**26.42.4.46 void jbox\_set\_hintstring (t\_object \* *bb*, char \* *s*)**

Set a box's hint text using a C-string.

**Parameters:**

- bb* The box to query.
- s* The new text to use for the box's hint.

**Returns:**

A Max error code.

**26.42.4.47 t\_max\_err jbox\_set\_ignoreclick (t\_object \* *b*, char *c*)**

Set whether a box ignores clicks.

**Parameters:**

- b* The box to query.
- c* 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.

**26.42.4.48 t\_max\_err jbox\_set\_outline (t\_object \* *b*, char *c*)**

Set whether a box draws an outline.

**Parameters:**

- b* The box to query.
- c* Pass zero to hide the outline, or non-zero to indicate that the box should draw the outline.

**Returns:**

A Max error code.

**26.42.4.49 t\_max\_err jbox\_set\_patching\_position (t\_object \* *box*, t\_pt \* *pos*)**

Set the position of a box for the patching view.

**Parameters:**

- box* The box whose position will be changed.
- pos* The address of a [t\\_pt](#) with the new x and y values.

**Returns:**

A Max error code.

**26.42.4.50 t\_max\_err jbox\_set\_patching\_rect (t\_object \* *box*, t\_rect \* *pr*)**

Change the patching rect of a box.

**Parameters:**

*box* The box whose rect will be changed.

*pr* The address of a [t\\_rect](#) with the new rect values.

**Returns:**

A Max error code.

**26.42.4.51 t\_max\_err jbox\_set\_patching\_size (t\_object \* *box*, t\_size \* *size*)**

Set the size of a box for the patching view.

**Parameters:**

*box* The box whose size will be changed.

*size* The address of a [t\\_size](#) with the new width and height values.

**Returns:**

A Max error code.

**26.42.4.52 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:**

*box* The box whose position will be changed.

*pos* The address of a [t\\_pt](#) with the new x and y values.

**Returns:**

A Max error code.

**26.42.4.53 t\_max\_err jbox\_set\_presentation (t\_object \* *b*, char *c*)**

Determine if a box is included in the presentation view.

**Parameters:**

*b* The box to query.

*c* 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.

**26.42.4.54 t\_max\_err jbox\_set\_presentation\_position (t\_object \* box, t\_pt \* pos)**

Set the position of a box for the presentation view.

**Parameters:**

*box* The box whose rect will be changed.

*pos* The address of a [t\\_pt](#) with the new x and y values.

**Returns:**

A Max error code.

**26.42.4.55 t\_max\_err jbox\_set\_presentation\_rect (t\_object \* box, t\_rect \* pr)**

Change the presentation rect of a box.

**Parameters:**

*box* The box whose rect will be changed.

*pr* The address of a [t\\_rect](#) with the new rect values.

**Returns:**

A Max error code.

**26.42.4.56 t\_max\_err jbox\_set\_presentation\_size (t\_object \* box, t\_size \* size)**

Set the size of a box for the presentation view.

**Parameters:**

*box* The box whose size will be changed.

*size* The address of a [t\\_size](#) with the new width and height values.

**Returns:**

A Max error code.

**26.42.4.57 t\_max\_err jbox\_set\_rect (t\_object \* box, t\_rect \* pr)**

Set both the presentation rect and the patching rect.

**Parameters:**

*box* The box whose rect will be changed.

*pr* The address of a [t\\_rect](#) with the new rect values.

**Returns:**

A Max error code.

**26.42.4.58 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:**

*box* The box whose rect will be changed.

*which* The name of the rect attribute to be changed, for example `_sym_presentation_rect` or `_sym_patching_rect`.

*pr* The address of a valid `t_rect` that will replace the current values used by the box.

**Returns:**

A Max error code.

**26.42.4.59 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:**

*box* The box whose rect will be changed.

*patcherview* A patcherview in which the box exists.

*rect* The address of a valid `t_rect` that will replace the current values used by the box in the given view.

**Returns:**

A Max error code.

**26.42.4.60 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:**

*box* The box whose size will be changed.

*size* The address of a `t_size` with the new size values.

**Returns:**

A Max error code.

**26.42.4.61 t\_max\_err jbox\_set\_varname (t\_object \* b, t\_symbol \* ps)**

Set a box's scripting name.

**Parameters:**

*b* The box to query.

*ps* The new scripting name for the box.

**Returns:**

A Max error code.

## 26.43 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.*



### 26.43.1 Detailed Description

A patch cord.

### 26.43.2 Function Documentation

#### 26.43.2.1 `t_object* jpatchline_get_box1 (t_object * l)`

Return the object box from which a patchline originates.

**Parameters:**

*l* A pointer to the patchline's instance.

**Returns:**

The object box from which the patchline originates.

#### 26.43.2.2 `t_object* jpatchline_get_box2 (t_object * l)`

Return the destination object box for a patchline.

**Parameters:**

*l* A pointer to the patchline's instance.

**Returns:**

The destination object box for a patchline.

#### 26.43.2.3 `t_max_err jpatchline_get_color (t_object * l, t_jrgba * prgba)`

Get the color of a patch line.

**Parameters:**

*l* A patchline instance.

*prgba* The address of a valid `t_jrgba` struct that will be filled with the color values of the patch line.

**Returns:**

An error code.

#### 26.43.2.4 `t_max_err jpatchline_get_endpoint (t_object * l, double * x, double * y)`

Retrieve a patchline's ending point.

**Parameters:**

*l* A pointer to the patchline's instance.

*x* The address of a variable to hold the x-coordinate of the ending point's position upon return.

*y* The address of a variable to hold the y-coordinate of the ending point's position upon return.

**Returns:**

A Max error code.

**26.43.2.5 char jpatchline\_get\_hidden (t\_object \* *l*)**

Determine if a patch line is hidden.

**Parameters:**

*l* A patchline instance.

**Returns:**

Zero if the patchline is visible, non-zero if it is hidden.

**26.43.2.6 long jpatchline\_get\_inletnum (t\_object \* *l*)**

Return the inlet number of the destination object box to which a patchline is connected.

**Parameters:**

*l* A pointer to the patchline's instance.

**Returns:**

The inlet number.

**26.43.2.7 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.

**26.43.2.8 long jpatchline\_get\_nummidpoints (t\_object \* *l*)**

Determine the number of midpoints (segments) in a patchline.

**Parameters:**

*l* A pointer to the patchline's instance.

**Returns:**

The number of midpoints in the patchline.

**26.43.2.9 long jpatchline\_get\_outletnum (t\_object \* *l*)**

Return the outlet number of the originating object box from which a patchline begins.

**Parameters:**

*l* A pointer to the patchline's instance.

**Returns:**

The outlet number.

**26.43.2.10 t\_max\_err jpatchline\_get\_startpoint (t\_object \* *l*, double \* *x*, double \* *y*)**

Retrieve a patchline's starting point.

**Parameters:**

*l* 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.

**26.43.2.11 t\_max\_err jpatchline\_set\_color (t\_object \* *l*, t\_jrgba \* *prgba*)**

Set the color of a patch line.

**Parameters:**

*l* A patchline instance.

*prgba* The address of a valid [t\\_jrgba](#) struct containing the color to use.

**Returns:**

An error code.

**26.43.2.12 t\_max\_err jpatchline\_set\_hidden (t\_object \* *l*, char *c*)**

Set a patchline's visibility.

**Parameters:**

*l* A patchline instance.

*c* Pass 0 to make a patchline visible, or non-zero to hide it.

**Returns:**

An error code.

## 26.44 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.*
- `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.*

### 26.44.1 Detailed Description

A view of a patcher.

### 26.44.2 Function Documentation

#### 26.44.2.1 `t_object* patcherview_findpatcherview (int x, int y)`

Find a patcherview at the given screen coords.

**Parameters:**

*x* The horizontal coordinate at which to find a patcherview.

*y* 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.

#### 26.44.2.2 `t_object* patcherview_get_jgraphics (t_object *pv)`

Given a patcherview, return the `t_jgraphics` context for that view.

**Parameters:**

*pv* The patcherview instance.

**Returns:**

The `t_jgraphics` context for the view.

#### 26.44.2.3 `char patcherview_get_locked (t_object *p)`

Find out if a patcherview is locked.

**Parameters:**

*p* The patcherview instance whose attribute value will be fetched.

**Returns:**

Returns 0 if unlocked, otherwise returns non-zero.

#### 26.44.2.4 `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:**

*pv* 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.

#### 26.44.2.5 `t_object* patcherview_get_patcher (t_object * pv)`

Given a patcherview, return its patcher.

**Parameters:**

*pv* The patcherview instance for which to fetch the patcher.

**Returns:**

The patcher.

#### 26.44.2.6 `char patcherview_get_presentation (t_object * pv)`

Find out if a patcherview is a presentation view.

**Parameters:**

*pv* The patcherview instance whose attribute value will be fetched.

**Returns:**

Returns 0 if the view is not a presentation view, otherwise returns non-zero.

#### 26.44.2.7 `t_max_err patcherview_get_rect (t_object * pv, t_rect * pr)`

Get the value of the rect attribute for a patcherview.

**Parameters:**

*pv* The patcherview instance whose attribute value will be fetched.

*pr* The address of a valid [t\\_rect](#) struct, whose contents will be filled upon return.

**Returns:**

An error code.

**26.44.2.8 char patcherview\_get\_visible (t\_object \* pv)**

Query a patcherview to determine whether it is visible.

**Parameters:**

*pv* The patcherview instance to query.

**Returns:**

Returns zero if the patcherview is invisible, otherwise returns non-zero.

**26.44.2.9 double patcherview\_get\_zoomfactor (t\_object \* pv)**

Fetch the zoom-factor of a patcherview.

**Parameters:**

*pv* The patcherview instance whose attribute value will be fetched.

**Returns:**

The factor by which the view is zoomed.

**26.44.2.10 t\_max\_err patcherview\_set\_locked (t\_object \* p, char c)**

Lock or unlock a patcherview.

**Parameters:**

*p* The patcherview instance whose attribute value will be set.

*c* Set this value to zero to unlock the patcherview, otherwise pass a non-zero value.

**Returns:**

An error code.

**26.44.2.11 t\_max\_err patcherview\_set\_presentation (t\_object \* p, char c)**

Set whether or not a patcherview is a presentation view.

**Parameters:**

*p* The patcherview instance whose attribute value will be set.

*c* Set this value to non-zero to make the patcherview a presentation view, otherwise pass zero.

**Returns:**

An error code.

**26.44.2.12 t\_max\_err patcherview\_set\_rect (t\_object \* *pv*, t\_rect \* *pr*)**

Set the value of the rect attribute for a patcherview.

**Parameters:**

*pv* The patcherview instance whose attribute value will be set.

*pr* The address of a valid [t\\_rect](#) struct.

**Returns:**

An error code.

**26.44.2.13 t\_max\_err patcherview\_set\_visible (t\_object \* *pv*, char *c*)**

Set the 'visible' attribute of a patcherview.

**Parameters:**

*pv* The patcherview instance whose attribute will be set.

*c* Whether or not the patcherview should be made visible.

**Returns:**

An error code.

**26.44.2.14 t\_max\_err patcherview\_set\_zoomfactor (t\_object \* *pv*, double *d*)**

Set the zoom-factor of a patcherview.

**Parameters:**

*pv* The patcherview instance whose attribute value will be set.

*d* The zoom-factor at which the patcherview should display the patcher.

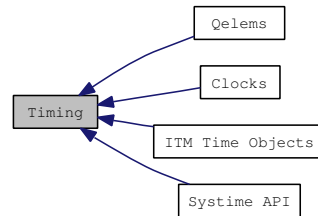
**Returns:**

An error code.



## 26.45 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.*

## 26.46 Clocks

Clock objects are your interface to Max's scheduler.

Collaboration diagram for Clocks:



### Typedefs

- typedef [t\\_object](#) [t\\_clock](#)  
A clock.

### Functions

- void \* [clock\\_new](#) (void \*obj, [method](#) fn)  
Create a new Clock object.
- void [clock\\_delay](#) (void \*x, long n)  
Schedule the execution of a Clock.
- void [clock\\_unset](#) (void \*x)  
Cancel the scheduled execution of a Clock.
- void [clock\\_fdelay](#) (void \*c, double time)  
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\\_object](#) \*x, void \*c, long time)  
Schedule a Clock on a scheduler.
- void [setclock\\_unset](#) ([t\\_object](#) \*x, void \*c)  
Remove a Clock from a scheduler.
- long [setclock\\_gettime](#) ([t\\_object](#) \*x)  
Find out the current time value of a setclock object.
- void [setclock\\_fdelay](#) ([t\\_object](#) \*s, void \*c, double time)  
Schedule a Clock on a scheduler, using a floating-point time argument.
- void [setclock\\_gettime](#) ([t\\_object](#) \*s, double \*time)  
Find out the current time value of a setclock object in floating-point milliseconds.
- double [systimer\\_gettime](#) (void)

While most Max/MSP 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.*
- void \* `scheduler_new` (void)  
*Create a new local scheduler.*
- void \* `scheduler_set` (void \*x)  
*Make a scheduler current, so that future related calls (such as `clock_delay()`) will affect the appropriate scheduler.*
- void `scheduler_run` (void \*x, double until)  
*Run scheduler events to a selected time.*
- void `scheduler_settime` (void \*x, double time)  
*Set the current time of the scheduler.*
- void `scheduler_gettime` (void \*x, double \*time)  
*Retrieve the current time of the selected scheduler.*

### 26.46.1 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.

### 26.46.2 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);
}

```

### 26.46.3 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.

### 26.46.3.1 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_ob.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);
}
}

```

## 26.46.4 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.

## 26.46.5 Function Documentation

### 26.46.5.1 void clock\_delay (void \* *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:

- x* Clock to schedule.
- n* Delay, in milliseconds, before the Clock will execute.

#### See also:

[clock\\_fdelay\(\)](#)

### 26.46.5.2 void clock\_fdelay (void \* *c*, double *time*)

Schedule the execution of a Clock using a floating-point argument. [clock\\_fdelay\(\)](#) sets a clock to go off at a certain number of milliseconds from the current logical time.

#### Parameters:

- c* Clock to schedule.
- time* Delay, in milliseconds, before the Clock will execute.

#### See also:

[clock\\_delay\(\)](#)

**26.46.5.3 void clock\_gettime (double \* *time*)**

Find out the current logical time of the scheduler in milliseconds as a floating-point number.

**Parameters:**

*time* Returns the current time.

**See also:**

[gettime\(\)](#)  
[setclock\\_gettime\(\)](#)  
[setclock\\_gettime\(\)](#)

**26.46.5.4 void\* 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:**

*obj* Argument that will be passed to clock function *fn* when it is called. This will almost always be a pointer to your object.

*fn* Function to be called when the clock goes off, declared to take a single argument as shown in [Using Clocks](#).

**Returns:**

A pointer to a newly created Clock object.

**26.46.5.5 void clock\_unset (void \* *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:**

*x* Clock to cancel.

**26.46.5.6 long gettime (void)**

Find out the current logical time of the scheduler in milliseconds.

**Returns:**

Returns the current time.

**See also:**

[clock\\_gettime\(\)](#)

**26.46.5.7 void scheduler\_gettime (void \* *x*, double \* *time*)**

Retrieve the current time of the selected scheduler.

**Parameters:**

- x* The scheduler to query.
- time* The current time of the selected scheduler.

**See also:**

[Creating Schedulers](#)

**26.46.5.8 void\* scheduler\_new (void)**

Create a new local scheduler.

**Returns:**

- A pointer to the newly created scheduler.

**See also:**

[Creating Schedulers](#)

**26.46.5.9 void scheduler\_run (void \* *x*, double *until*)**

Run scheduler events to a selected time.

**Parameters:**

- x* The scheduler to advance.
- until* The ending time for this run (in milliseconds).

**See also:**

[Creating Schedulers](#)

**26.46.5.10 void\* scheduler\_set (void \* *x*)**

Make a scheduler current, so that future related calls (such as [clock\\_delay\(\)](#)) will affect the appropriate scheduler.

**Parameters:**

- x* 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](#)



**26.46.5.11 void scheduler\_settime (void \* *x*, double *time*)**

Set the current time of the scheduler.

**Parameters:**

*x* The scheduler to set.

*time* The new current time for the selected scheduler (in milliseconds).

**See also:**

[Creating Schedulers](#)

**26.46.5.12 void setclock\_delay (t\_object \* *x*, void \* *c*, long *time*)**

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:**

*x* A setclock object to be used for scheduling this clock.

*c* Clock object containing the function to be executed.

*time* 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\(\)](#)

**26.46.5.13 void setclock\_fdelay (t\_object \* *s*, void \* *c*, double *time*)**

Schedule a Clock on a scheduler, using a floating-point time argument.

**Parameters:**

*s* A setclock object to be used for scheduling this clock.

*c* Clock object containing the function to be executed.

*time* 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\(\)](#)

**26.46.5.14 void setclock\_gettime (t\_object \* *s*, double \* *time*)**

Find out the current time value of a setclock object in floating-point milliseconds.

**Parameters:**

- s* A setclock object.  
*time* The current time in milliseconds.

**See also:**

[Scheduling with setclock Objects](#)  
[setclock\\_gettime\(\)](#)

**26.46.5.15 long setclock\_gettime (t\_object \* x)**

Find out the current time value of a setclock object.

**Parameters:**

- x* 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\(\)](#)

**26.46.5.16 void setclock\_unset (t\_object \* x, void \* 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](#)

**26.46.5.17 double systimer\_gettime (void)**

While most Max/MSP 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.

## 26.47 Qelems

Your object's methods may be called at interrupt level.

Collaboration diagram for Qelems:



### Typedefs

- typedef void \* [t\\_qelem](#)  
A *qelem*.

### Functions

- void \* [qelem\\_new](#) (void \*obj, [method](#) fn)  
Create a new *Qelem*.
- void [qelem\\_set](#) (void \*q)  
Cause a *Qelem* to execute.
- void [qelem\\_unset](#) (void \*q)  
Cancel a *Qelem*'s execution.
- void [qelem\\_free](#) (void \*x)  
Free a *Qelem* object created with [qelem\\_new\(\)](#).
- void [qelem\\_front](#) (void \*x)  
Cause a *Qelem* to execute with a higher priority.

#### 26.47.1 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.

## 26.47.2 Function Documentation

### 26.47.2.1 void qelem\_free (void \* *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.

### 26.47.2.2 void qelem\_front (void \* *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.

### 26.47.2.3 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 *fn* 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.

**26.47.2.4 void qelem\_set (void \* *q*)**

Cause a Qelem to execute.

**Parameters:**

*q* 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.

**26.47.2.5 void qelem\_unset (void \* *q*)**

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:**

*q* The Qelem whose execution you wish to cancel.

## 26.48 Systeime API

The Systeime API provides the means of getting the system time, instead of the scheduler time as you would with functions like [gettime\(\)](#).

Collaboration diagram for Systeime API:



### Data Structures

- struct [t\\_datetime](#)

*The Systeime data structure.*

### Enumerations

- enum [e\\_max\\_dateflags](#) {  
    [SYSDATEFORMAT\\_FLAGS\\_SHORT](#) = 1,  
    [SYSDATEFORMAT\\_FLAGS\\_MEDIUM](#) = 2,  
    [SYSDATEFORMAT\\_FLAGS\\_LONG](#) = 3 }

*Flags for the [sysdateformat\\_formatdatetime\(\)](#) function.*

### Functions

- unsigned long [systeime\\_ticks](#) (void)  
*Find out the operating system's time in ticks.*
- unsigned long [systeime\\_ms](#) (void)  
*Find out the operating system's time in milliseconds.*
- void [systeime\\_datetime](#) ([t\\_datetime](#) \*d)  
*Find out the operating system's date and time.*
- unsigned long [systeime\\_seconds](#) (void)  
*Find out the operating system's time in seconds.*
- void [systeime\\_secondstodate](#) (unsigned long secs, [t\\_datetime](#) \*d)  
*Convert a time in seconds into a [t\\_datetime](#) representation.*
- unsigned long [systeime\\_datetoseconds](#) ([t\\_datetime](#) \*d)  
*Convert a [t\\_datetime](#) representation of time into seconds.*
- void [sysdateformat\\_strftimeodatetime](#) (char \*strf, [t\\_datetime](#) \*d)  
*Fill a [t\\_datetime](#) struct with a datetime formatted string.*

- void `sysdateformat_formatdatetime` (`t_datetime` \*`d`, long `dateflags`, long `timeflags`, char \*`s`, long `buflen`)

*Get a human friendly string representation of a `t_datetime`.*

## 26.48.1 Detailed Description

The Systeime API provides the means of getting the system time, instead of the scheduler time as you would with functions like `gettime()`.

## 26.48.2 Enumeration Type Documentation

### 26.48.2.1 enum `e_max_dateflags`

Flags for the `sysdateformat_formatdatetime()` function.

**Enumerator:**

`SYSDATEFORMAT_FLAGS_SHORT` short  
`SYSDATEFORMAT_FLAGS_MEDIUM` medium  
`SYSDATEFORMAT_FLAGS_LONG` long

Definition at line 34 of file `ext_systeime.h`.

## 26.48.3 Function Documentation

### 26.48.3.1 void `sysdateformat_formatdatetime` (`t_datetime` \*`d`, long `dateflags`, long `timeflags`, char \*`s`, long `buflen`)

Get a human friendly string representation of a `t_datetime`. For example: "Today", "Yesterday", etc.

**Parameters:**

`d` The address of a `t_datetime` to fill.  
`dateflags` One of the values defined in `e_max_dateflags`.  
`timeflags` Currently unused. Pass 0.  
`s` An already allocated string to hold the human friendly result.  
`buflen` The number of characters allocated to the string `s`.

### 26.48.3.2 void `sysdateformat_strftimetodatetime` (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:**

`strf` A string containing the datetime.  
`d` The address of a `t_datetime` to fill.

**26.48.3.3 void systime\_datetime (t\_datetime \* d)**

Find out the operating system's date and time.

**Parameters:**

*d* Returns the system's date and time in a [t\\_datetime](#) data structure.

**26.48.3.4 unsigned long systime\_datetoseconds (t\_datetime \* d)**

Convert a [t\\_datetime](#) representation of time into seconds.

**Parameters:**

*d* The address of a [t\\_datetime](#) that contains a valid period of time.

**Returns:**

The number of seconds represented by *d*.

Referenced by db\_util\_stringtodate().

**26.48.3.5 unsigned long systime\_ms (void)**

Find out the operating system's time in milliseconds.

**Returns:**

the system time in milliseconds.

**26.48.3.6 unsigned long systime\_seconds (void)**

Find out the operating system's time in seconds.

**Returns:**

the system time in seconds.

**26.48.3.7 void systime\_secondstodate (unsigned long secs, t\_datetime \* d)**

Convert a time in seconds into a [t\\_datetime](#) representation.

**Parameters:**

*secs* A number of seconds to be represented as a [t\\_datetime](#).

*d* The address of a [t\\_datetime](#) that will be filled with the converted value.

Referenced by db\_util\_datetostring().



**26.48.3.8 unsigned long systime\_ticks (void)**

Find out the operating system's time in ticks.

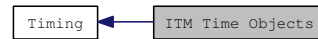
**Returns:**

the system time in ticks.

## 26.49 ITM Time Objects

ITM Time Objects are a high-level interface to ITM, a tempo-based scheduler API.

Collaboration diagram for ITM Time Objects:



### Enumerations

- enum {
  - `TIME_FLAGS_LOCATION` = 1,
  - `TIME_FLAGS_TICKSONLY` = 2,
  - `TIME_FLAGS_FIXEDONLY` = 4,
  - `TIME_FLAGS_LOOKAHEAD` = 8,
  - `TIME_FLAGS_USECLOCK` = 16,
  - `TIME_FLAGS_USEQELEM` = 32,
  - `TIME_FLAGS_FIXED` = 64,
  - `TIME_FLAGS_PERMANENT` = 128,
  - `TIME_FLAGS_TRANSPORT` = 256,
  - `TIME_FLAGS_EVENTLIST` = 512,
  - `TIME_FLAGS_CHECKSCHEDULE` = 1024,
  - `TIME_FLAGS_LISTENTICKS` = 2048,
  - `TIME_FLAGS_NOUNITS` = 4096,
  - `TIME_FLAGS_BBUSOURCE` = 8192,
  - `TIME_FLAGS_POSITIVE` = 16384 }

*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, 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 its 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)`  
*Convert a time value in milliseconds to the equivalent value in ticks, given the context of a specified itm object.*
- 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.*
- 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 itm object.*
- void `itm_barbeatunitstoticks (t_itm *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 itm object.*

- void `itm_tickstobarbeatunits` (`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)  
*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, char \*attrname, 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_itm`  
*A low-level object for tempo-based scheduling.*
- BEGIN\_USING\_C\_LINKAGE typedef `t_object` `t_timeobject`  
*A high-level time object for tempo-based scheduling.*

### 26.49.1 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).

### 26.49.2 Enumeration Type Documentation

#### 26.49.2.1 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 0

Definition at line 29 of file ext\_itm.h.

### 26.49.3 Function Documentation

#### 26.49.3.1 void class\_time\_addattr (t\_class \* *c*, char \* *attrname*, char \* *attrlabel*, long *flags*)

Create an attribute permitting a time object to be changed in a user-friendly way.

##### Parameters:

*c* Class being initialized.

*attrname* Name of the attribute associated with the time object.

*attrlabel* Descriptive label for the attribute (appears in the inspector)

*flags* Options, see "Flags that determine time object behavior" above

#### 26.49.3.2 void itm\_barbeatunitstoticks (t\_itm \* *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 itm object.

##### Parameters:

*x* An itm object.

*bars* The measure number of the location/position.

*beats* The beat number of the location/position.

*units* The number of ticks past the beat of the location/position.

*ticks* The address of a variable to hold the number of ticks upon return.

*position* Set this parameter to [TIME\\_FLAGS\\_LOCATION](#) or to zero (for position mode).

#### 26.49.3.3 void itm\_dereference (t\_itm \* *x*)

Stop referencing an itm object. When you are done using an itm object, you must call this function to decrement its reference count.

##### Parameters:

*x* The itm object.

**26.49.3.4 void itm\_dump (t\_itm \* x)**

Print diagnostic information about an itm object to the Max window.

**Parameters:**

*x* The itm object.

**26.49.3.5 void\* itm\_getglobal (void)**

Return the global (default / unnamed) itm object.

**Returns:**

The global [t\\_itm](#) object.

**26.49.3.6 t\_symbol\* itm\_getname (t\_itm \* x)**

Given an itm object, return its name.

**Parameters:**

*x* The itm object.

**Returns:**

The name of the itm.

**26.49.3.7 void\* itm\_getnamed (t\_symbol \* s, long create)**

Return a named itm object.

**Parameters:**

*s* The name of the itm to return.

*create* If non-zero, then create this named itm should it not already exist.

**Returns:**

The global [t\\_itm](#) object.

**26.49.3.8 double itm\_getresolution (void)**

Get the number of ticks-per-quarter-note globally from the itm system.

**Returns:**

The number of ticks-per-quarter-note.

**See also:**

[itm\\_setresolution\(\)](#)

### 26.49.3.9 long itm\_getstate (t\_itm \* x)

Find out if time is currently progressing for a given itm object.

**Parameters:**

*x* The itm object.

**Returns:**

Returns non-zero if the time is running, or zero if it is paused.

**See also:**

[itm\\_pause\(\)](#)

[itm\\_resume\(\)](#)

### 26.49.3.10 double itm\_getticks (t\_itm \* x)

Report the current time of the itm in ticks. You can use functions such as [itm\\_tickstobarbeatunits\(\)](#) or [itm\\_tickstoms\(\)](#) to convert to a different representation of the time.

**Parameters:**

*x* The itm object.

**Returns:**

The current time in ticks.

### 26.49.3.11 double itm\_gettime (t\_itm \* x)

Report the current internal time. This is the same as calling [clock\\_gettime\(\)](#);

**Parameters:**

*x* The itm object.

**Returns:**

The current internal time.

### 26.49.3.12 void itm\_gettimesignature (t\_itm \* x, long \* num, long \* denom)

Query an itm object for its current time signature.

**Parameters:**

*x* The itm object.

*num* The address of a variable to hold the top number of the time signature upon return.

*denom* The address of a variable to hold the bottom number of the time signature upon return.



**26.49.3.13 long itm\_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:**

*u* 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).

**26.49.3.14 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:**

*x* An itm object.

*ms* A time specified in ms.

**Returns:**

The time specified in samples.

**26.49.3.15 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:**

*x* An itm object.

*ms* A time specified in ms.

**Returns:**

The time specified in ticks.

**26.49.3.16 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 itm object.

**26.49.3.17 void itm\_reference (t\_itm \* *x*)**

Reference an itm object. When you are using an itm object, you must call this function to increment its reference count.

**Parameters:**

*x* The itm object.

**26.49.3.18 void itm\_resume (t\_itm \* *x*)**

Start the passage of time for an itm 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 itm object.

**26.49.3.19 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 itm object.

**Parameters:**

*x* An itm object.

*samps* A time specified in samples.

**Returns:**

The time specified in ms.

**26.49.3.20 void itm\_setresolution (double *res*)**

Set the number of ticks-per-quarter-note globally for the itm system. The default is 480.

**Parameters:**

*res* The number of ticks-per-quarter-note.

**See also:**

[itm\\_getresolution\(\)](#)

**26.49.3.21 void itm\_settimesignature (t\_itm \* *x*, long *num*, long *denom*, long *flags*)**

Set an itm object's current time signature.

**Parameters:**

*x* The itm object.

*num* The top number of the time signature.

*denom* The bottom number of the time signature.

*flags* Currently unused -- pass zero.

**26.49.3.22 void itm\_tickstobarbeatunits (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.

**Parameters:**

*x* An itm object.

*ticks* The number of ticks to translate into a time represented as bars, beats, and ticks.

*bars* The address of a variable to hold the measure number of the location/position upon return.

*beats* The address of a variable to hold the beat number of the location/position upon return.

*units* The address of a variable to hold the number of ticks past the beat of the location/position upon return.

*position* Set this parameter to [TIME\\_FLAGS\\_LOCATION](#) or to zero (for position mode).

**26.49.3.23 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.

**Parameters:**

*x* An itm object.

*ticks* A time specified in ticks.

**Returns:**

The time specified in ms.

**26.49.3.24 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:**

*ox* Time object.

*vitm* The associated ITM object (use [time\\_getitm\(\)](#) to determine it).

*oq* A time object that holds a quantization interval, can be NULL.

**Returns:**

Interval (in ticks) for scheduling this object.

**26.49.3.25 void\* time\_getitm (t\_timeobject \* *ox*)**

Return the ITM object associated with this time object.

**Parameters:**

*ox* Time object.

**Returns:**

The associated [t\\_itm](#) object.

**26.49.3.26 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.

**26.49.3.27 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.

**26.49.3.28 void time\_getphase (t\_timeobject \* *tx*, double \* *phase*, double \* *slope*)**

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.

**26.49.3.29 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.

**26.49.3.30 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.

**26.49.3.31 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:**

*x* The time object.

*attr* Name of the millisecond based attribute in the owning object that will be updated

*flags* If TIME\_FLAGS\_LISTENTICKS is passed here, updating will not happen if the time value is fixed (ms) based

**26.49.3.32 void\* time\_new (t\_object \* owner, t\_symbol \* attrname, method tick, long flags)**

Create a new time object.

**Parameters:**

*owner* Object that will own this time object (task routine, if any, will pass owner as argument).

*attrname* Name of the attribute associated with the time object.

*tick* Task routine that will be executed (can be NULL)

*flags* Options, see "Flags that determine time object behavior" above

**Returns:**

The newly created [t\\_timeobject](#).

**26.49.3.33 void time\_now (t\_timeobject \* x, t\_timeobject \* quantize)**

Schedule a task for right now, with optional quantization.

**Parameters:**

*x* The time object that schedules temporary events. The time interval is ignored and 0 ticks is used instead.

*quantize* A time object that holds a quantization interval, can be NULL for no quantization.

**26.49.3.34 void time\_schedule (t\_timeobject \* *x*, t\_timeobject \* *quantize*)**

Schedule a task, with optional quantization.

**Parameters:**

- x* The time object that schedules temporary events (must have been created with TIME\_FLAGS\_USECLOCK but not TIME\_FLAGS\_PERMANENT)
- quantize* A time object that holds a quantization interval, can be NULL for no quantization.

**26.49.3.35 void time\_schedule\_limit (t\_timeobject \* *x*, t\_timeobject \* *quantize*)**

Schedule a task, with optional minimum interval,.

**Parameters:**

- x* The time object that schedules temporary events (must have been created with TIME\_FLAGS\_USECLOCK but not TIME\_FLAGS\_PERMANENT)
- quantize* The minimum interval into the future when the event can occur, can be NULL if there is no minimum interval.

**26.49.3.36 void time\_setclock (t\_timeobject \* *tx*, t\_symbol \* *sc*)**

Associate a named setclock object with a time object (unsupported).

**Parameters:**

- tx* Time object.
- sc* Name of an associated setclock object.

**26.49.3.37 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:**

- tx* The time object.
- s* Message selector.
- argc* Count of arguments.
- argv* Message arguments.

**26.49.3.38 void time\_stop (t\_timeobject \* *x*)**

Stop a currently scheduled time object.

**Parameters:**

- x* The time object.

**26.49.3.39 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:**

*x* The time object.

**26.49.4 Variable Documentation****26.49.4.1 BEGIN\_USING\_C\_LINKAGE typedef t\_object t\_itm**

A low-level object for tempo-based scheduling.

**See also:**

[t\\_timeobject](#)  
[ITM](#)

Definition at line 13 of file ext\_itm.h.

**26.49.4.2 BEGIN\_USING\_C\_LINKAGE typedef t\_object t\_timeobject**

A high-level time object for tempo-based scheduling.

**See also:**

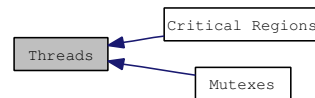
[t\\_itm](#)  
[ITM](#)

Definition at line 24 of file ext\_time.h.

## 26.50 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](#)

### Defines

- `#define ATOMIC_INCREMENT(pv) (_InterlockedIncrement(pv))`

*Use this routine for incrementing a global counter using a threadsafe and multiprocessor safe method.*

- `#define ATOMIC_DECREMENT(pv) (_InterlockedDecrement(pv))`

*Use this routine for decrementing a global counter using a threadsafe and multiprocessor safe method.*

### Typedefs

- `typedef void t_thread`

*A Max thread.*

- `typedef void * t_systhread`

*An opaque thread instance pointer.*

- `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 = 0x00000000,`  
`SYSTHREAD_MUTEX_ERRORCHECK = 0x00000001,`  
`SYSTHREAD_MUTEX_RECURSIVE = 0x00000002 }`



*`systhread_mutex_new()` flags*

## Functions

- void `schedule` (void \*ob, method fun, long when, `t_symbol` \*sym, short argc, 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()`.*
- `t_systhread` `systhread_self` (void)  
*Return the thread instance pointer for the calling 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 the scheduler thread.*

### 26.50.1 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/JitterSdkSchedQueue>  
<http://www.cycling74.com/story/2005/5/2/133649/9742>

### 26.50.2 Define Documentation

#### 26.50.2.1 `#define ATOMIC_DECREMENT(pv) (_InterlockedDecrement(pv))`

Use this routine for decrementing a global counter using a threadsafe and multiprocessor safe method.

**Parameters:**

*pv* pointer to the (int) counter.

Definition at line 33 of file ext\_atomic.h.

#### 26.50.2.2 `#define ATOMIC_INCREMENT(pv) (_InterlockedIncrement(pv))`

Use this routine for incrementing a global counter using a threadsafe and multiprocessor safe method.

**Parameters:**

*pv* pointer to the (int) counter.

Definition at line 26 of file ext\_atomic.h.

### 26.50.3 Enumeration Type Documentation

#### 26.50.3.1 `enum e_max_systhread_mutex_flags`

[systhread\\_mutex\\_new\(\)](#) flags

**Enumerator:**

***SYSTHREAD\_MUTEX\_NORMAL*** Normal.

***SYSTHREAD\_MUTEX\_ERRORCHECK*** Error-checking.

***SYSTHREAD\_MUTEX\_RECURSIVE*** Recursive.

Definition at line 29 of file ext\_systhread.h.

## 26.50.4 Function Documentation

### 26.50.4.1 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:

- ob* First argument passed to the function fun when it executes.
- fn* Function to be called, see below for how it should be declared.
- sym* Second argument passed to the function fun when it executes.
- argc* Count of arguments in argv. argc is also the third argument passed to the function fun when it executes.
- argv* Array containing a variable number of [t\\_atom](#) 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\(\)](#)

### 26.50.4.2 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:

- ob* First argument passed to the function fun when it executes.
- fn* Function to be called, see below for how it should be declared.
- sym* Second argument passed to the function fun when it executes.
- argc* Count of arguments in argv. argc is also the third argument passed to the function fun when it executes.
- argv* Array containing a variable number of [t\\_atom](#) 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\(\)](#)

**26.50.4.3 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.

**26.50.4.4 void schedule (void \*ob, method fun, long when, t\_symbol \*sym, short argc, Atom \*argv)**

Cause a function to be executed at the timer level at some time in the future.

**Parameters:**

**ob** First argument passed to the function *fun* when it executes.

**fun** Function to be called, see below for how it should be declared.

**when** The logical time that the function *fun* will be executed.

**sym** Second argument passed to the function *fun* when it executes.

**argc** Count of arguments in *argv*. *argc* is also the third argument passed to the function *fun* when it executes.

**argv** Array containing a variable number of [t\\_atom](#) 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.

**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\(\)](#)

#### 26.50.4.5 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:**

***ob*** First argument passed to the function *fun* when it executes.

***fun*** Function to be called, see below for how it should be declared.

***delay*** The delay from the current time before the function will be executed.

***sym*** Second argument passed to the function *fun* when it executes.

***argc*** Count of arguments in *argv*. *argc* is also the third argument passed to the function *fun* when it executes.

***argv*** Array containing a variable number of [t\\_atom](#) 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.

**Remarks:**

[schedule\\_delay\(\)](#) is similar to `schedule` but allows you to specify the time as a delay rather than a specific logical time.

One use of [schedule\(\)](#) or [schedule\\_delay\(\)](#) is as an alternative to using the lockout flag. Here is an example click method that calls [schedule\(\)](#) instead of [outlet\\_int\(\)](#) surrounded by `lockout_set()` calls.

```
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\(\)](#)

#### 26.50.4.6 long `systhread_create` (method *entryproc*, void \* *arg*, long *stacksize*, long *priority*, long *flags*, t\_systhread \* *thread*)

Create a new thread.

**Parameters:**

- entryproc* A method to call in the new thread when the thread is created.
- arg* An argument to pass to the method specified for *entryproc*. Typically this might be a pointer to your object's struct.
- stacksize* Not used. Pass 0 for this argument.
- priority* Not used. Pass 0 for this argument.
- flags* Not used. Pass 0 for this argument.
- thread* The address of a [t\\_systhread](#) where this thread's instance pointer will be stored.

**Returns:**

A Max error code as defined in [e\\_max\\_errorcodes](#).

**26.50.4.7 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:**

- status* You will typically pass 0 for status. This value will be accessible by [systhread\\_join\(\)](#), if needed.

**26.50.4.8 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.

**26.50.4.9 short systhread\_istimerthread (void)**

Check to see if the function currently being executed is in the scheduler thread.

**Returns:**

Returns true if the function is being executed in the main thread, otherwise false.

**26.50.4.10 long systhread\_join (t\_systhread thread, unsigned int \* retval)**

Wait for thread to quit and get return value from [systhread\\_exit\(\)](#).

**Parameters:**

- thread* The thread to join.
- retval* The address of a long to hold the return value (status) from [systhread\\_exit\(\)](#).

**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.

**26.50.4.11 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.

**26.50.4.12 void systhread\_sleep (long *milliseconds*)**

Suspend the execution of the calling thread.

**Parameters:**

*milliseconds* The number of milliseconds to suspend the execution of the calling thread. The actual amount of time may be longer depending on various factors.

**26.50.4.13 long systhread\_terminate (t\_systhread *thread*)**

Forcefully kill a thread -- not recommended.

**Parameters:**

*thread* The thread to kill.

**Returns:**

A Max error code as defined in [e\\_max\\_errorcodes](#).

## 26.51 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:



### Typedefs

- typedef void [t\\_critical](#)  
*A Max critical region.*

### 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.*

#### 26.51.1 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".

## 26.51.2 Function Documentation

### 26.51.2.1 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 [t\\_critical](#) struct, or zero to uses Max's global critical region.

**See also:**

[critical\\_exit\(\)](#)

### 26.51.2.2 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, you 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 [t\\_critical](#) struct, or zero to uses Max's global critical region.

### 26.51.2.3 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 [t\\_critical](#) struct that will be freed.

### 26.51.2.4 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 [t\\_critical](#) struct will be returned to you via this pointer.

### 26.51.2.5 short critical\_tryenter (t\_critical *x*)

Try to enter a critical region if it is not locked.

**Parameters:**

*x* A pointer to a [t\\_critical](#) struct, or zero to uses Max's global critical region.

**Returns:**

returns non-zero if there was a problem entering

**See also:**

[critical\\_enter\(\)](#)

## 26.52 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\(\)](#).*

### 26.52.1 Detailed Description

See also:

[Critical Regions](#)

### 26.52.2 Function Documentation

#### 26.52.2.1 long systhread\_mutex\_free (t\_systhread\_mutex pmutex)

Free a mutex created with [systhread\\_mutex\\_new\(\)](#).

**Parameters:**

*pmutex* The mutex instance pointer.

**Returns:**

A Max error code as defined in [e\\_max\\_errorcodes](#).

### 26.52.2.2 long `systhread_mutex_lock` (`t_systhread_mutex pmutex`)

Enter block of locked code until a `systhread_mutex_unlock()` is reached. It is important to keep the code in this block as small as possible.

#### Parameters:

*pmutex* The mutex instance pointer.

#### Returns:

A Max error code as defined in `e_max_errorcodes`.

#### See also:

`systhread_mutex_trylock()`

### 26.52.2.3 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:

*pmutex* The address of a variable to store the mutex pointer.

*flags* Flags to determine the behaviour of the mutex, as defined in `e_max_systhread_mutex_flags`.

#### 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.

### 26.52.2.4 long `systhread_mutex_newlock` (`t_systhread_mutex * pmutex`, `long flags`)

Convenience utility that combines `systhread_mutex_new()` and `systhread_mutex_lock()`.

#### Parameters:

*pmutex* The address of a variable to store the mutex pointer.

*flags* Flags to determine the behaviour of the mutex, as defined in `e_max_systhread_mutex_flags`.

#### Returns:

A Max error code as defined in `e_max_errorcodes`.

**26.52.2.5 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:**

*pmutex* The mutex instance pointer.

**Returns:**

Returns non-zero if there was a problem entering.

**See also:**

[systhread\\_mutex\\_lock\(\)](#)

**26.52.2.6 long systhread\_mutex\_unlock (t\_systhread\_mutex *pmutex*)**

Exit a block of code locked with [systhread\\_mutex\\_lock\(\)](#).

**Parameters:**

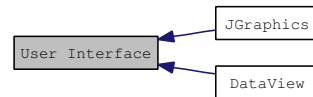
*pmutex* The mutex instance pointer.

**Returns:**

A Max error code as defined in [e\\_max\\_errorcodes](#).

## 26.53 User Interface

Collaboration diagram for User Interface:



### Modules

- [JGraphics](#)

*The JGraphics API provided in Max 5 is an interface for drawing based on the API for the Cairo vector graphics library ( [http://en.wikipedia.org/wiki/Cairo\\_%28graphics%29](http://en.wikipedia.org/wiki/Cairo_%28graphics%29), <http://cairographics.org/manual/> ).*

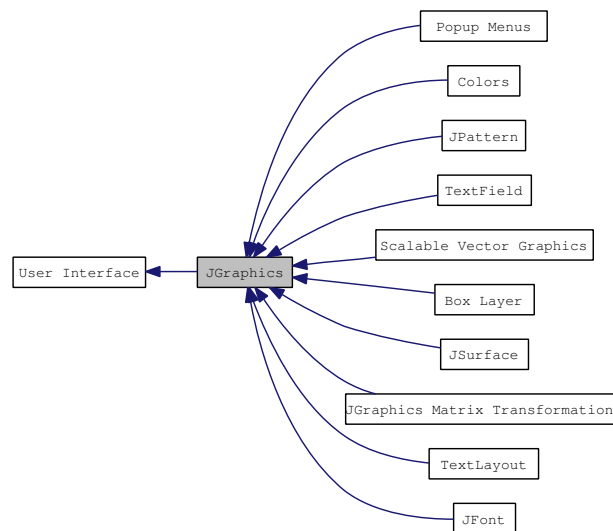
- [DataView](#)

*The jdataview object provides a mechanism to display data in a tabular format.*

## 26.54 JGraphics

The JGraphics API provided in Max 5 is an interface for drawing based on the API for the Cairo vector graphics library ( [http://en.wikipedia.org/wiki/Cairo\\_%28graphics%29](http://en.wikipedia.org/wiki/Cairo_%28graphics%29) , <http://cairographics.org/manual/> ).

Collaboration diagram for JGraphics:



### 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.*

### 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](#)
- [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.*

## Defines

- `#define JGRAPHICS_RECT_BOTTOM(rect) (((rect)->y)+((rect)->height))`  
*Determine the coordinate of the bottom of a rect.*
- `#define JGRAPHICS_RECT_RIGHT(rect) (((rect)->x)+((rect)->width))`  
*Determine the coordinate of the right side of a rect.*
- `#define JGRAPHICS_PI (3.1415926535897932384626433832795028842)`  
*Utility macro to return the value of Pi.*
- `#define JGRAPHICS_2PI (2. * 3.1415926535897932384626433832795028842)`  
*Utility macro to return the value of twice Pi.*
- `#define JGRAPHICS_PIOVER2 (0.5 * 3.1415926535897932384626433832795028842)`  
*Utility macro to return the value of half of Pi.*
- `#define JGRAPHICS_3PIOVER2 ((2.0 * JGRAPHICS_PI) / 3.0)`  
*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 _jsvg t_jsvg`  
*An instance of an SVG object.*
- `typedef struct _jpopupmenu t_jpopupmenu`  
*An instance of a pop-up menu.*



## 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` = 1,  
    `JGRAPHICS_TEXT_JUSTIFICATION_RIGHT` = 2,  
    `JGRAPHICS_TEXT_JUSTIFICATION_HCENTERED` = 4,  
    `JGRAPHICS_TEXT_JUSTIFICATION_TOP` = 8,  
    `JGRAPHICS_TEXT_JUSTIFICATION_BOTTOM` = 16,  
    `JGRAPHICS_TEXT_JUSTIFICATION_VCENTERED` = 32,  
    `JGRAPHICS_TEXT_JUSTIFICATION_HJUSTIFIED` = 64,  
    `JGRAPHICS_TEXT_JUSTIFICATION_CENTERED` = `JGRAPHICS_TEXT_JUSTIFICATION_HCENTERED` + `JGRAPHICS_TEXT_JUSTIFICATION_VCENTERED` }  
*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.*
- 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.*
- 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_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_jfont` (`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)  
*Display text at the current position in a context.*
- 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_measure_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_canantialias_text_to_transparent_bg` ()  
*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, long \*\*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.

### 26.54.1 Detailed Description

The JGraphics API provided in Max 5 is an interface for drawing based on the API for the Cairo vector graphics library ( [http://en.wikipedia.org/wiki/Cairo\\_%28graphics%29](http://en.wikipedia.org/wiki/Cairo_%28graphics%29) , <http://cairographics.org/manual/> ). Internally, the drawing is rendered using JUCE ( <http://rawmaterialsoftware.com/juce/> ), however JUCE functions cannot be called directly.

### 26.54.2 Define Documentation

#### 26.54.2.1 `#define JGRAPHICS_2PI (2. * 3.1415926535897932384626433832795028842)`

Utility macro to return the value of twice Pi.

Definition at line 65 of file jgraphics.h.

#### 26.54.2.2 `#define JGRAPHICS_3PIOVER2 ((2.0 * JGRAPHICS_PI) / 3.0)`

Utility macro to return the 270° Case.

Definition at line 71 of file jgraphics.h.

#### 26.54.2.3 `#define JGRAPHICS_PI (3.1415926535897932384626433832795028842)`

Utility macro to return the value of Pi.

Definition at line 62 of file jgraphics.h.

#### 26.54.2.4 `#define JGRAPHICS_PIOVER2 (0.5 * 3.1415926535897932384626433832795028842)`

Utility macro to return the value of half of Pi.

Definition at line 68 of file jgraphics.h.

### 26.54.3 Enumeration Type Documentation

#### 26.54.3.1 `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.

Definition at line 95 of file jgraphics.h.

### 26.54.3.2 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.

Definition at line 85 of file jgraphics.h.

### 26.54.3.3 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.

Definition at line 900 of file jgraphics.h.

## 26.54.4 Function Documentation

### 26.54.4.1 void jgraphics\_append\_path (t\_jgraphics \* g, t\_jpath \* path)

Add a path to a graphics context.

**Parameters:**

*g* The graphics context.

*path* The path to add.

### 26.54.4.2 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:**

*g* The graphics context.

*xc* The horizontal coordinate of the arc's center.  
*yc* The vertical coordinate of the arc's center.  
*radius* The radius of the arc.  
*angle1* The starting angle of the arc in radians. Zero radians is center right (positive x axis).  
*angle2* The terminal angle of the arc in radians. Zero radians is center right (positive x axis).

#### 26.54.4.3 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:

*g* The graphics context.  
*xc* The horizontal coordinate of the arc's center.  
*yc* The vertical coordinate of the arc's center.  
*radius* The radius of the arc.  
*angle1* The starting angle of the arc in radians. Zero radians is center right (positive x axis).  
*angle2* The terminal angle of the arc in radians. Zero radians is center right (positive x axis).

#### 26.54.4.4 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:

*g* The graphics context.

#### 26.54.4.5 t\_jpath\* jgraphics\_copy\_path (t\_jgraphics \* g)

Get a copy of the current path from a context.

##### Parameters:

*g* A copy of the current path.

#### 26.54.4.6 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:

*g* The graphics context.  
*x1* The first control point.  
*y1* The first control point.  
*x2* The second control point.  
*y2* The second control point.  
*x3* The destination point.  
*y3* The destination point.

**26.54.4.7 void jgraphics\_destroy (t\_jgraphics \* g)**

Release or free a graphics context.

**Parameters:**

*g* The context to release.

**26.54.4.8 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.

**26.54.4.9 void jgraphics\_ellipse (t\_jgraphics \* g, double x, double y, double width, double height)**

Add a closed elliptical path in the context.

**Parameters:**

*g* The graphics context.

*x* The horizontal origin.

*y* The vertical origin.

*width* The width of the rect.

*height* The height of the rect.

**26.54.4.10 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:**

*g* Pointer to a jgraphics context.

*extents* The address of a [t\\_jgraphics\\_font\\_extents](#) structure to be filled with the results.

**26.54.4.11 void jgraphics\_get\_current\_point (t\_jgraphics \* g, double \* x, double \* y)**

Get the current location of the cursor in a graphics context.

**Parameters:**

*g* The graphics context.

*x* The address of a variable that will be set to the horizontal cursor location upon return.

*y* The address of a variable that will be set to the vertical cursor location upon return.

**26.54.4.12 void jgraphics\_getfiletypes (void \* *dummy*, long \* *count*, long \*\* *filetypes*, char \* *alloc*)**

Get a list of filetypes appropriate for use with jgraphics surfaces.

**Parameters:**

*dummy* Unused.

*count* The address of a variable to be set with the number of types in filetypes upon return.

*filetypes* The address of a variable that will represent the array of file types upon return.

*alloc* 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];
long      *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)
    system_free_ptr((char *)type);
```

**26.54.4.13 void jgraphics\_line\_to (t\_jgraphics \* *g*, double *x*, double *y*)**

Add a line segment to the current path.

**Parameters:**

*g* The graphics context.

*x* The destination point.

*y* The destination point.

**26.54.4.14 void jgraphics\_move\_to (t\_jgraphics \* *g*, double *x*, double *y*)**

Move the cursor to a new point and begin a new subpath.



**Parameters:**

- g* The graphics context.
- x* The new location.
- y* The new location.

**26.54.4.15 void jgraphics\_new\_path (t\_jgraphics \* *g*)**

Begin a new path. This action clears any current path in the context.

**Parameters:**

- g* The graphics context.

**26.54.4.16 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:**

- g* The graphics context.
- x* The horizontal origin.
- y* The vertical origin.
- width* The width of the oval.
- height* The height of the oval.

**26.54.4.17 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:**

- g* The graphics context.
- xc* The horizontal coordinate of the arc's center.
- yc* The vertical coordinate of the arc's center.
- radiusx* The horizontal radius of the arc.
- radiusy* The vertical radius of the arc.
- angle1* The starting angle of the arc in radians. Zero radians is center right (positive x axis).
- angle2* The terminal angle of the arc in radians. Zero radians is center right (positive x axis).

**26.54.4.18 void jgraphics\_path\_destroy (t\_jpath \* *path*)**

Release/free a path.

**Parameters:**

- path* The path to release.

**26.54.4.19 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:**

*g* The graphics context.  
*cornerRadius* The amount by which to round corners.

**26.54.4.20 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:**

*positioned\_rect* The address of a valid [t\\_rect](#) whose members will be filled in upon return.  
*positioned\_near\_this\_rect* A pointer to a rect near which this rect should be positioned.  
*keep\_inside\_this\_rect* A pointer to a rect defining the limits within which the new rect must reside.

**26.54.4.21 void jgraphics\_rectangle (t\_jgraphics \* g, double x, double y, double width, double height)**

Add a closed rectangle path in the context.

**Parameters:**

*g* The graphics context.  
*x* The horizontal origin.  
*y* The vertical origin.  
*width* The width of the rect.  
*height* The height of the rect.

**26.54.4.22 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:**

*g* The graphics context.  
*x* The horizontal origin.  
*y* The vertical origin.  
*width* The width of the rect.  
*height* The height of the rect.  
*ovalwidth* The width of the oval used for the round corners.  
*ovalheight* The height of the oval used for the round corners.

**26.54.4.23 long jgraphics\_rectcontainsrect (t\_rect \* outer, t\_rect \* inner)**

Simple utility to test for rectangle containment.

**Parameters:**

- outer* The address of the first rect for the test.
- inner* The address of the second rect for the test.

**Returns:**

Returns true if the inner rect is completely inside the outer rect, otherwise false.

**26.54.4.24 long jgraphics\_rectintersectsrect (t\_rect \* r1, t\_rect \* r2)**

Simple utility to test for rectangle intersection.

**Parameters:**

- r1* The address of the first rect for the test.
- r2* The address of the second rect for the test.

**Returns:**

Returns true if the rects intersect, otherwise false.

**26.54.4.25 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:**

- g* The context you wish to reference.

**Returns:**

A new reference to the context.

**26.54.4.26 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:**

- g* The graphics context.
- x1* The first control point.
- y1* The first control point.
- x2* The second control point.
- y2* The second control point.
- x3* The destination point.
- y3* The destination point.

**26.54.4.27 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:**

- g* The graphics context.
- x* The destination point.
- y* The destination point.

**26.54.4.28 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:**

- g* The graphics context.
- x* The new location.
- y* The new location.

**26.54.4.29 int jgraphics\_round (double d)**

Utility for rounding a double to an int.

**Parameters:**

- d* floating-point input.

**Returns:**

- rounded int output.

**26.54.4.30 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:**

- g* The graphics context.
- family* The name of the font family (e.g. "Arial").
- slant* Define the slant to use for the font.
- weight* Define the weight to use for the font.

**26.54.4.31 void jgraphics\_select\_jfont (t\_jgraphics \* g, t\_jfont \* jfont)**

Specify a font for a graphics context by passing a [t\\_jfont](#) object.

**Parameters:**

- g* The graphics context.
- jfont* A jfont object whose attributes will be copied to the context.

**26.54.4.32 void jgraphics\_set\_font\_size (t\_jgraphics \* *g*, double *size*)**

Specify the font size for a context.

**Parameters:**

*g* The graphics context.  
*size* The font size.

**26.54.4.33 void jgraphics\_set\_underline (t\_jgraphics \* *g*, char *underline*)**

Turn underlining on/off for text in a context.

**Parameters:**

*g* The graphics context.  
*underline* Pass true or false to set the appropriate behavior.

**26.54.4.34 void jgraphics\_show\_text (t\_jgraphics \* *g*, const char \* *utf8*)**

Display text at the current position in a context.

**Parameters:**

*g* The graphics context.  
*utf8* The text to display.

**26.54.4.35 long jgraphics\_system\_canantialias texttotransparentbg ()**

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.

**26.54.4.36 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:**

*g* Pointer to a jgraphics context.  
*utf8* A string containing the text whose dimensions we wish to find.  
*width* The address of a variable to be filled with the width of the rendered text.  
*height* The address of a variable to be filled with the height of the rendered text.

**26.54.4.37** `void jgraphics_text_measure_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:**

*g* Pointer to a jgraphics context.

*utf8* A string containing the text whose dimensions we wish to find.

*wrapwidth* The number of pixels in width at which the text should be wrapped if it is too long.

*includewhitespace* Set zero to not include white space in the calculation, otherwise set this parameter to 1.

*width* The address of a variable to be filled with the width of the rendered text.

*height* The address of a variable to be filled with the height of the rendered text.

*numlines* The address of a variable to be filled with the number of lines required to render the text.

**26.54.4.38** `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.

## 26.55 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.*

- `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)`
- `const unsigned char * jgraphics_image_surface_lockpixels_readonly (t_jsurface *s, int x, int y, int width, int height, int *linestride, int *pixelstride)`  
*Get a read-only raw bitmap of a surface.*
- `void jgraphics_image_surface_unlockpixels_readonly (t_jsurface *s, const unsigned char *data)`  
*Unlock a surface locked by `jgraphics_image_surface_lockpixels_readonly()`.*
- `unsigned char * jgraphics_image_surface_lockpixels (t_jsurface *s, int x, int y, int width, int height, int *linestride, int *pixelstride)`  
*Get a writable bitmap of a surface.*
- `void jgraphics_image_surface_unlockpixels (t_jsurface *s, unsigned char *data)`  
*Unlock a surface locked by `jgraphics_image_surface_lockpixels()`.*
- `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.*

### 26.55.1 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.



## 26.55.2 Function Documentation

### 26.55.2.1 `t_jgraphics* jgraphics_create (t_jsurface * target)`

Create a context to draw on a particular surface. When you are done, call [jgraphics\\_destroy\(\)](#).

**Parameters:**

*target* The surface to which to draw.

**Returns:**

The new graphics context.

### 26.55.2.2 `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:**

*moduleRef* A pointer to your external's module, which is passed to your external's main() function when the class is loaded.

*resname* Base name of the resource data (without an extension)

*extcount* Count of possible extensions (ignored on Windows)

*exts* Array of symbol atoms containing possible filename extensions (ignored on Windows)

*data* Returned resource data assigned to a pointer you supply

*datasize* Size of the data returned

**Remarks:**

You are responsible for freeing any data returned in the data pointer

**Returns:**

A Max error code.

### 26.55.2.3 `void jgraphics_image_surface_clear (t_jsurface * s, int x, int y, int width, int height)`

Set all pixels in rect to 0.

**Parameters:**

*s* The surface to clear.

*x* The horizontal origin of the rect to clear.

*y* The vertical origin of the rect to clear.

*width* The width of the rect to clear.

*height* The height of the rect to clear.

#### 26.55.2.4 **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:

*format* Defines the color format for the new surface.  
*width* Defines the width of the new surface.  
*height* Defines the height of the new surface.

##### Returns:

A pointer to the new surface.

#### 26.55.2.5 **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:

*data* The data. For example, an RGBA image loaded in memory.  
*format* The format of the data.  
*width* The width of the new surface.  
*height* The height of the new surface.  
*stride* The number of bytes between the start of rows in the data buffer.  
*freefun* If not NULL, freefun will be called when the surface is destroyed  
*freearg* This will be passed to freefun if/when freefun is called.

##### Returns:

A pointer to the new surface.

#### 26.55.2.6 **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:

*filename* The name of the file.  
*path* The path id of the file.

##### Returns:

A pointer to the new surface.

**26.55.2.7 t\_jsurface\* jgraphics\_image\_surface\_create\_from\_filedata (const void \* *data*, unsigned long *datalen*)**

Create a new surface from file data.

**Parameters:**

*data* A pointer to the raw PNG or JPG bits.

*datalen* The number of bytes in data.

**Returns:**

The new surface.

**See also:**

[jgraphics\\_write\\_image\\_surface\\_to\\_filedata\(\)](#)

**26.55.2.8 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:**

*moduleRef* A pointer to your external's module, which is passed to your external's main() function when the class is loaded.

*resname* 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;
}
```

**26.55.2.9 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:**

*filename* The name of the file.

*path* The path id of the file.

**Returns:**

A pointer to the new surface.

**26.55.2.10 void jgraphics\_image\_surface\_draw (t\_jgraphics \* g, t\_jsurface \* s, t\_rect srcRect, t\_rect destRect)**

Draw an image surface. This 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:**

*g* The graphics context in which to draw the surface.

*s* The surface to draw.

*srcRect* The rect within the surface that should be drawn.

*destRect* The rect in the context to which to draw the srcRect.

**See also:**

[jgraphics\\_image\\_surface\\_draw\\_fast\(\)](#)

**26.55.2.11 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:**

*g* The graphics context in which to draw the surface.

*s* The surface to draw.

**See also:**

[jgraphics\\_image\\_surface\\_draw](#)

**26.55.2.12 int jgraphics\_image\_surface\_get\_height (t\_jsurface \* s)**

Retrieve the height of a surface.

**Parameters:**

*s* The surface to query.

**Returns:**

The height of the surface.

**26.55.2.13 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:**

- s* The surface.
- x* The horizontal coordinate of the pixel.
- y* The vertical coordinate of the pixel.
- color* The address of a valid [t\\_jrgba](#) struct whose values will be filled in with the color of the pixel upon return.

**26.55.2.14 int jgraphics\_image\_surface\_get\_width (t\_jsurface \* s)**

Retrieve the width of a surface.

**Parameters:**

- s* The surface to query.

**Returns:**

The width of the surface.

**26.55.2.15 unsigned char\* jgraphics\_image\_surface\_lockpixels (t\_jsurface \* s, int x, int y, int width, int height, int \* linestyle, int \* pixelstride)**

Get a writable bitmap of a surface. After you are done reading/writing to the bitmap, you should call [jgraphics\\_image\\_surface\\_unlockpixels\(\)](#).

**Parameters:**

- s* The surface.
- x* The rect horizontal-origin for the raw bitmap.
- y* The rect vertical-origin for the raw bitmap.
- width* The rect width for the bitmap.
- height* The rect height for the bitmap.
- linestyle* The line stride for the bitmap.
- pixelstride* The pixel stride for the bitmap.

**Returns:**

A pointer to the raw bitmap.

**26.55.2.16 const unsigned char\* jgraphics\_image\_surface\_lockpixels\_readonly (t\_jsurface \* s, int x, int y, int width, int height, int \* linestyle, int \* pixelstride)**

Get a read-only raw bitmap of a surface. This gives you a pointer to the raw bitmap data so you can read from it yourself and examine the stuff that was drawn. The [jgraphics\\_image\\_surface\\_lockpixels\(\)](#) version gives you a pointer to the bits that you can directly munge. After either call you should make the appropriate unlock call.

**Parameters:**

- s* The surface.
- x* The rect horizontal-origin for the raw bitmap.
- y* The rect vertical-origin for the raw bitmap.
- width* The rect width for the bitmap.
- height* The rect height for the bitmap.
- linestride* The line stride for the bitmap.
- pixelstride* The pixel stride for the bitmap.

**Returns:**

A pointer to the raw bitmap.

**26.55.2.17** `void jgraphics_image_surface_scroll (t_jsurface * s, int x, int y, int width, int height, int dx, int dy, t_jpath ** path)`

**Parameters:**

- s* The surface to scroll.
- x* The origin of the rect to scroll.
- y* The origin of the rect to scroll.
- width* The width of the rect to scroll.
- height* The height of the rect to scroll.
- dx* The amount to scroll the surface horizontally.
- dy* The amount to scroll the surface vertically.
- path* 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.

**26.55.2.18** `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:**

- s* The surface.
- x* The horizontal coordinate of the pixel.
- y* The vertical coordinate of the pixel.
- color* The color of the pixel.

**26.55.2.19** `void jgraphics_image_surface_unlockpixels (t_jsurface * s, unsigned char * data)`

Unlock a surface locked by [jgraphics\\_image\\_surface\\_lockpixels\(\)](#).

**Parameters:**

- s* The surface.
- data* The pointer returned by [jgraphics\\_image\\_surface\\_lockpixels\(\)](#).

**26.55.2.20** void `jgraphics_image_surface_unlockpixels_readonly` (`t_jsurface * s`, const unsigned char \* *data*)

Unlock a surface locked by `jgraphics_image_surface_lockpixels_readonly()`.

**Parameters:**

*s* The surface.

*data* The pointer returned by `jgraphics_image_surface_lockpixels_readonly()`.

**26.55.2.21** 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:**

*surface* The surface to export.

*filename* Specify the name of the file to create.

*path* Specify the path id for where to create the file.

*dpi* Define the resolution of the image (e.g. 72).

**Returns:**

A Max error code.

**26.55.2.22** void `jgraphics_surface_destroy` (`t_jsurface * s`)

Release or free a surface.

**Parameters:**

*s* The surface to release.

**26.55.2.23** 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:**

*s* The surface to reference.

**Returns:**

The new reference to the surface.

### 26.55.2.24 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:

*surf* The surface whose data will be retrieved.

*fmt* The format for the data. This should be a selection from [t\\_jgraphics\\_fileformat](#).

*data* The address of a pointer that will be allocated and filled. When you are done with this data you should free it using [systemem\\_freeptr\(\)](#).

*size* 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;
}
```

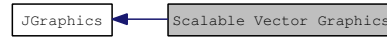
#### See also:

[jgraphics\\_image\\_surface\\_create\\_from\\_filedata\(\)](#)



## 26.56 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.

### 26.56.1 Function Documentation

#### 26.56.1.1 `t_jsvg* jsvg_create_from_file` (const char \*filename, short path)

Read an SVG file, return a `t_jsvg` object.

##### Parameters:

*filename* The name of the file to read.

*path* The path id of the file to read.

##### Returns:

A new SVG object.

#### 26.56.1.2 `t_jsvg* jsvg_create_from_resource` (const void \* moduleRef, const char \* resname)

Read an SVG file from a resource.

##### Parameters:

*moduleRef* The external's moduleRef.

*resname* The name of the SVG resource.

**Returns:**

A new SVG object.

**See also:**

[jgraphics\\_image\\_surface\\_create\\_from\\_resource\(\)](#)

**26.56.1.3 t\_jsvg\* jsvg\_create\_from\_xmlstring (const char \* *svgXML*)**

Create an SVG object from a string containing the SVG's XML.

**Parameters:**

*svgXML* The SVG source.

**Returns:**

A new SVG object.

**26.56.1.4 void jsvg\_destroy (t\_jsvg \* *svg*)**

Free a [t\\_jsvg](#) object.

**Parameters:**

*svg* The object to free.

**26.56.1.5 void jsvg\_get\_size (t\_jsvg \* *svg*, double \* *width*, double \* *height*)**

Retrieve the size of an SVG object.

**Parameters:**

*svg* An SVG object.

*width* The address of a variable that will be set to the width upon return.

*height* The address of a variable that will be set to the width upon return.

**26.56.1.6 void jsvg\_render (t\_jsvg \* *svg*, t\_jgraphics \* *g*)**

Render an SVG into a graphics context.

**Parameters:**

*svg* The SVG object to render.

*g* The graphics context in which to render.

## 26.57 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\\_reference](#) ([t\\_jfont](#) \*font)  
*Create new reference to an existing font object.*
- void [jfont\\_destroy](#) ([t\\_jfont](#) \*font)  
*Release or free a font object.*
- void [jfont\\_set\\_font\\_size](#) ([t\\_jfont](#) \*font, double size)  
*Set the size of a font object.*
- void [jfont\\_set\\_underline](#) ([t\\_jfont](#) \*font, char ul)  
*Set the underlining 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\\_measure\\_wrapped](#) ([t\\_jfont](#) \*font, const char \*utf8, double wrapwidth, long include-whitespace, 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.*
- [t\\_max\\_err jfont\\_getfontlist](#) (long \*count, [t\\_symbol](#) \*\*\*list)  
*Get a list of font names.*
- 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.*

## 26.57.1 Enumeration Type Documentation

### 26.57.1.1 enum [t\\_jgraphics\\_font\\_slant](#)

Enumeration of slanting options for font display.

#### Enumerator:

***JGRAPHICS\_FONT\_SLANT\_NORMAL*** Normal slanting (typically this means no slanting).  
***JGRAPHICS\_FONT\_SLANT\_ITALIC*** Italic slanting.

Definition at line 663 of file [jgraphics.h](#).

### 26.57.1.2 enum [t\\_jgraphics\\_font\\_weight](#)

Enumeration of font weight options for font display.

#### Enumerator:

***JGRAPHICS\_FONT\_WEIGHT\_NORMAL*** Normal font weight.  
***JGRAPHICS\_FONT\_WEIGHT\_BOLD*** Bold font weight.

Definition at line 672 of file [jgraphics.h](#).

## 26.57.2 Function Documentation

### 26.57.2.1 long [jbox\\_get\\_font\\_slant](#) ([t\\_object](#) \* b)

Get the slant box's font.

#### Parameters:

***b*** An object's box.

**Returns:**

A value from the `t_jgraphics_font_slant` enum.

**26.57.2.2 long jbox\_get\_font\_weight (t\_object \* *b*)**

Get the slant box's font.

**Parameters:**

*b* An object's box.

**Returns:**

A value from the `t_jgraphics_font_weight` enum.

**26.57.2.3 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:**

*family* The name of the font family (e.g. Arial).

*slant* The type of slant for the font.

*weight* The type of weight for the font.

*size* The size of the font.

**Returns:**

The new font object.

**26.57.2.4 void jfont\_destroy (t\_jfont \* *font*)**

Release or free a font object.

**Parameters:**

*font* The font object to release.

**26.57.2.5 void jfont\_extents (t\_jfont \* *font*, t\_jgraphics\_font\_extents \* *extents*)**

Get extents of this font.

**Parameters:**

*font* The font object.

*extents* The font extents upon return/

**26.57.2.6 t\_max\_err jfont\_getfontlist (long \* *count*, t\_symbol \*\*\* *list*)**

Get a list of font names.

**Parameters:**

*count* The address of a variable to hold the count of font names in list upon return.

*list* The address of a [t\\_symbol\\*\\*](#) initialized to NULL. Upon return this will be set to an array of count [t\\_symbol](#) pointers. This array should be freed using [sysmem\\_freeptr\(\)](#) when you are done with it.

**Returns:**

A Max error code.

**26.57.2.7 t\_jfont\* jfont\_reference (t\_jfont \* *font*)**

Create new reference to an existing font object.

**Parameters:**

*font* The font object for which to obtain a reference.

**Returns:**

The new font object reference.

**26.57.2.8 void jfont\_set\_font\_size (t\_jfont \* *font*, double *size*)**

Set the size of a font object.

**Parameters:**

*font* The font object.

*size* The new size for the font object.

**26.57.2.9 void jfont\_set\_underline (t\_jfont \* *font*, char *ul*)**

Set the underlining of a font object.

**Parameters:**

*font* The font object.

*ul* Pass true to underline, or false for no underlining.

**26.57.2.10 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:**

*font* The font object.

*utf8* The text whose rendering will be measured.

*width* The address of a variable to hold the width upon return.

*height* The address of a variable to hold the height upon return.

**26.57.2.11 void jfont\_text\_measure\_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:**

*font* The font object.

*utf8* The text whose rendering will be measured.

*wrapwidth* The maximum width, above which text should wrap onto a new line.

*includewhitespace* If non-zero, include whitespace in the measurement.

*width* The address of a variable to hold the width upon return.

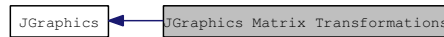
*height* The address of a variable to hold the height upon return.

*numlines* The address of a variable to hold the number of lines of text after wrapping upon return.

## 26.58 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.*



### 26.58.1 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.

### 26.58.2 Function Documentation

#### 26.58.2.1 `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;
```

#### 26.58.2.2 `void jgraphics_matrix_init_identity (t_jmatrix * x)`

Modify a matrix to be an identity transform.

##### Parameters:

*x* The [t\\_jmatrix](#).

#### 26.58.2.3 `void jgraphics_matrix_init_rotate (t_jmatrix * x, double radians)`

Initialize a [t\\_jmatrix](#) to rotate (offset) a point.

##### Parameters:

*x* The [t\\_jmatrix](#).  
*radians* The angle or rotation in radians.

**26.58.2.4 void jgraphics\_matrix\_init\_scale (t\_jmatrix \* *x*, double *sx*, double *sy*)**

Initialize a [t\\_jmatrix](#) to scale (offset) a point.

**Parameters:**

- x* The [t\\_jmatrix](#).
- sx* The horizontal scale factor.
- sy* The vertical scale factor.

**26.58.2.5 void jgraphics\_matrix\_init\_translate (t\_jmatrix \* *x*, double *tx*, double *ty*)**

Initialize a [t\\_jmatrix](#) to translate (offset) a point.

**Parameters:**

- x* The [t\\_jmatrix](#).
- tx* The amount of x-axis translation.
- ty* The amount of y-axis translation.

**26.58.2.6 void jgraphics\_matrix\_invert (t\_jmatrix \* *x*)**

Invert an existing matrix.

**Parameters:**

- x* The [t\\_jmatrix](#).

**26.58.2.7 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:**

- result* The resulting product [t\\_jmatrix](#).
- a* The first operand.
- b* The second operand.

**26.58.2.8 void jgraphics\_matrix\_rotate (t\_jmatrix \* *x*, double *radians*)**

Apply a rotation to an existing matrix.

**Parameters:**

- x* The [t\\_jmatrix](#).
- radians* The angle or rotation in radians.

**26.58.2.9 void jgraphics\_matrix\_scale (t\_jmatrix \* *x*, double *sx*, double *sy*)**

Apply a scaling to an existing matrix.

**Parameters:**

- x* The [t\\_jmatrix](#).
- sx* The horizontal scale factor.
- sy* The vertical scale factor.

**26.58.2.10 void jgraphics\_matrix\_transform\_point (const t\_jmatrix \* *matrix*, double \* *x*, double \* *y*)**

Transform a point using a [t\\_jmatrix](#) transformation.

**Parameters:**

- matrix* The [t\\_jmatrix](#).
- x* The address of the variable holding the x coordinate.
- y* The address of the variable holding the y coordinate.

**26.58.2.11 void jgraphics\_matrix\_translate (t\_jmatrix \* *x*, double *tx*, double *ty*)**

Apply a translation to an existing matrix.

**Parameters:**

- x* The [t\\_jmatrix](#).
- tx* The amount of x-axis translation.
- ty* The amount of y-axis translation.

## 26.59 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.*

#### 26.59.1 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()`.

## 26.60 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.*

### 26.60.1 Function Documentation

#### 26.60.1.1 `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:**

- argc* The number of atoms in the array provided in *argv*. This should be 3 or 4 depending on whether or not the alpha channel is being provided.
- argv* The address to the first of an array of atoms that define the color.
- c* The address of a [t\\_jrgba](#) struct for which the color will be defined.

**Returns:**

A Max error code.

**26.60.1.2 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:**

- jrgba* The color struct whose color will be retrieved.
- argc* 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.
- argv* The address of a [t\\_atom](#) 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 [system\\_freeptr\(\)](#).

**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.

**26.60.1.3 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:**

- jrgba* The color struct whose color will be set.
- argc* The number of atoms in the array. This must be 4.
- argv* 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.

**26.60.1.4 long jrgba\_compare (t\_jrgba \*rgba1, t\_jrgba \*rgba2)**

Compare two colors for equality.

**Parameters:**

*rgba1* The address of a `t_jrgba` struct to compare.

*rgba2* The address of another `t_jrgba` struct to compare.

**Returns:**

returns 1 if `rgba1 == rgba2`.

**26.60.1.5 void jrgba\_copy (t\_jrgba \* dest, t\_jrgba \* src)**

Copy a color.

**Parameters:**

*dest* The address of a `t_jrgba` struct to which the color will be copied.

*src* The address of a `t_jrgba` struct from which the color will be copied.

**26.60.1.6 void jrgba\_set (t\_jrgba \* prgba, double r, double g, double b, double a)**

Set the components of a color.

**Parameters:**

*prgba* The address of a `t_jrgba` struct for which the color will be defined.

*r* The value of the red component in a range of [0.0, 1.0].

*g* The value of the green component in a range of [0.0, 1.0].

*b* The value of the blue component in a range of [0.0, 1.0].

*a* The value of the alpha component in a range of [0.0, 1.0].

**26.60.1.7 void jrgba\_to\_atoms (t\_jrgba \* c, t\_atom \* argv)**

Get the components of a color in an array of pre-allocated atoms.

**Parameters:**

*argv* 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.

*c* The address of a `t_jrgba` struct from which the color components will be fetched.

## 26.61 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)  
*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.*

### 26.61.1 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](#)

### 26.61.2 Function Documentation

#### 26.61.2.1 `char textfield_get_autoscroll (t_object * tf)`

Return the value of the 'autoscroll' attribute of a textfield.

**Parameters:**

*tf* The textfield instance pointer.

**Returns:**

A value of the attribute.

#### 26.61.2.2 `t_max_err textfield_get_bgcolor (t_object * tf, t_jrgba * prgba)`

Retrieve the background color of a textfield.

**Parameters:**

*tf* The textfield instance pointer.

*prgba* The address of a valid [t\\_jrgba](#) whose values will be filled-in upon return.

**Returns:**

A Max error code.

#### 26.61.2.3 `char textfield_get_editonclick (t_object * tf)`

Return the value of the 'editonclick' attribute of a textfield.

**Parameters:**

*tf* The textfield instance pointer.

**Returns:**

A value of the attribute.

#### 26.61.2.4 `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:**

*tf* The textfield instance pointer.

**Returns:**

The current text used as the empty text.

**26.61.2.5 char textfield\_get\_noactivate (t\_object \* *tf*)**

Return the value of the 'noactivate' attribute of a textfield.

**Parameters:**

*tf* The textfield instance pointer.

**Returns:**

A value of the attribute.

**26.61.2.6 t\_object\* textfield\_get\_owner (t\_object \* *tf*)**

Return the object that owns a particular textfield.

**Parameters:**

*tf* The textfield instance pointer.

**Returns:**

A pointer to the owning object.

**26.61.2.7 char textfield\_get\_readonly (t\_object \* *tf*)**

Return the value of the 'readonly' attribute of a textfield.

**Parameters:**

*tf* The textfield instance pointer.

**Returns:**

A value of the attribute.

**26.61.2.8 char textfield\_get\_selectallonedit (t\_object \* *tf*)**

Return the value of the 'selectallonedit' attribute of a textfield.

**Parameters:**

*tf* The textfield instance pointer.

**Returns:**

A value of the attribute.

**26.61.2.9 t\_max\_err textfield\_get\_textcolor (t\_object \* *tf*, t\_jrgba \* *prgba*)**

Retrieve the color of the text in a textfield.

**Parameters:**

- tf* The textfield instance pointer.
- prgba* The address of a valid [t\\_jrgba](#) whose values will be filled-in upon return.

**Returns:**

A Max error code.

**26.61.2.10 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:**

- tf* The textfield instance pointer.
- pleft* The address of a variable to hold the value of the left margin upon return.
- ptop* The address of a variable to hold the value of the top margin upon return.
- pright* The address of a variable to hold the value of the right margin upon return.
- pbottom* The address of a variable to hold the value of the bottom margin upon return.

**Returns:**

A Max error code.

**26.61.2.11 char textfield\_get\_underline (t\_object \* *tf*)**

Return the value of the 'underline' attribute of a textfield.

**Parameters:**

- tf* The textfield instance pointer.

**Returns:**

A value of the attribute.

**26.61.2.12 char textfield\_get\_useellipsis (t\_object \* *tf*)**

Return the value of the 'useellipsis' attribute of a textfield.

**Parameters:**

- tf* The textfield instance pointer.

**Returns:**

A value of the attribute.

**26.61.2.13 char textfield\_get\_wantsreturn (t\_object \* *tf*)**

Return the value of the 'wantsreturn' attribute of a textfield.

**Parameters:**

*tf* The textfield instance pointer.

**Returns:**

A value of the attribute.

**26.61.2.14 char textfield\_get\_wantstab (t\_object \* *tf*)**

Return the value of the 'wantstab' attribute of a textfield.

**Parameters:**

*tf* The textfield instance pointer.

**Returns:**

A value of the attribute.

**26.61.2.15 char textfield\_get\_wordwrap (t\_object \* *tf*)**

Return the value of the 'wordwrap' attribute of a textfield.

**Parameters:**

*tf* The textfield instance pointer.

**Returns:**

A value of the attribute.

**26.61.2.16 t\_max\_err textfield\_set\_autoscroll (t\_object \* *tf*, char *c*)**

Set the 'autoscroll' attribute of a textfield.

**Parameters:**

*tf* The textfield instance pointer.

*c* The new value for the attribute.

**Returns:**

A Max error code.

**26.61.2.17 t\_max\_err textfield\_set\_bgcolor (t\_object \* *tf*, t\_jrgba \* *prgba*)**

Set the background color of a textfield.

**Parameters:**

- tf* The textfield instance pointer.
- prgba* The address of a [t\\_jrgba](#) containing the new color to use.

**Returns:**

A Max error code.

**26.61.2.18 t\_max\_err textfield\_set\_editonclick (t\_object \* *tf*, char *c*)**

Set the 'editonclick' attribute of a textfield.

**Parameters:**

- tf* The textfield instance pointer.
- c* The new value for the attribute.

**Returns:**

A Max error code.

**26.61.2.19 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:**

- tf* The textfield instance pointer.
- txt* A symbol containing the new text to display when the textfield has no content.

**Returns:**

A Max error code.

**26.61.2.20 t\_max\_err textfield\_set\_noactivate (t\_object \* *tf*, char *c*)**

Set the 'noactivate' attribute of a textfield.

**Parameters:**

- tf* The textfield instance pointer.
- c* The new value for the attribute.

**Returns:**

A Max error code.

**26.61.2.21 t\_max\_err textfield\_set\_readonly (t\_object \* *tf*, char *c*)**

Set the 'readonly' attribute of a textfield.

**Parameters:**

- tf* The textfield instance pointer.
- c* The new value for the attribute.

**Returns:**

A Max error code.

**26.61.2.22 t\_max\_err textfield\_set\_selectallonedit (t\_object \* *tf*, char *c*)**

Set the 'selectallonedit' attribute of a textfield.

**Parameters:**

- tf* The textfield instance pointer.
- c* The new value for the attribute.

**Returns:**

A Max error code.

**26.61.2.23 t\_max\_err textfield\_set\_textcolor (t\_object \* *tf*, t\_jrgba \* *prgba*)**

Set the color of the text in a textfield.

**Parameters:**

- tf* The textfield instance pointer.
- prgba* The address of a [t\\_jrgba](#) containing the new color to use.

**Returns:**

A Max error code.

**26.61.2.24 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:**

- tf* The textfield instance pointer.
- left* The new value for the left margin.
- top* The new value for the top margin.
- right* The new value for the right margin.
- bottom* The new value for the bottom margin.

**Returns:**

A Max error code.

**26.61.2.25 t\_max\_err textfield\_set\_underline (t\_object \* *tf*, char *c*)**

Set the 'underline' attribute of a textfield.

**Parameters:**

- tf* The textfield instance pointer.
- c* The new value for the attribute.

**Returns:**

A Max error code.

**26.61.2.26 t\_max\_err textfield\_set\_useellipsis (t\_object \* *tf*, char *c*)**

Set the 'useellipsis' attribute of a textfield.

**Parameters:**

- tf* The textfield instance pointer.
- c* The new value for the attribute.

**Returns:**

A Max error code.

**26.61.2.27 t\_max\_err textfield\_set\_wantsreturn (t\_object \* *tf*, char *c*)**

Set the 'wantsreturn' attribute of a textfield.

**Parameters:**

- tf* The textfield instance pointer.
- c* The new value for the attribute.

**Returns:**

A Max error code.

**26.61.2.28 t\_max\_err textfield\_set\_wantstab (t\_object \* *tf*, char *c*)**

Set the 'wantstab' attribute of a textfield.

**Parameters:**

- tf* The textfield instance pointer.
- c* The new value for the attribute.

**Returns:**

A Max error code.



**26.61.2.29 t\_max\_err textfield\_set\_wordwrap (t\_object \* *tf*, char *c*)**

Set the 'wordwrap' attribute of a textfield.

**Parameters:**

- tf* The textfield instance pointer.
- c* The new value for the attribute.

**Returns:**

A Max error code.

## 26.62 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](#) = 1,  
[JGRAPHICS\\_TEXTLAYOUT\\_USEELLIPSIS](#) = 3 }

*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\\_settextcolor](#) ([t\\_jtextlayout \\*textlayout](#), [t\\_jrgba \\*textcolor](#))  
*Set the color to render text in a textlayout object.*
- void [jtextlayout\\_measure](#) ([t\\_jtextlayout \\*textlayout](#), long startindex, long numchars, long include-whitespace, 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.*

### 26.62.1 Detailed Description

A textlayout is lower-level text rendering object used by higher-level entities such as [TextField](#).

### 26.62.2 Enumeration Type Documentation

#### 26.62.2.1 enum t\_jgraphics\_textlayout\_flags

Flags for setting text layout options.

##### Enumerator:

**JGRAPHICS\_TEXTLAYOUT\_NOWRAP** disable [word](#) wrapping

**JGRAPHICS\_TEXTLAYOUT\_USEELLIPSIS** show ... if a line doesn't fit (implies NOWRAP too)

Definition at line 914 of file jgraphics.h.

### 26.62.3 Function Documentation

#### 26.62.3.1 t\_jtextlayout\* jtextlayout\_create ()

Create a new textlayout object.

##### Returns:

The new textlayout object.

#### 26.62.3.2 void jtextlayout\_destroy (t\_jtextlayout \* *textlayout*)

Release/free a textlayout object.

##### Parameters:

*textlayout* The textlayout object to release.

#### 26.62.3.3 void jtextlayout\_draw (t\_jtextlayout \* *tl*, t\_jgraphics \* *g*)

Draw a textlayout in a given graphics context.

##### Parameters:

*tl* The textlayout object to query.

*g* The graphics context in which to draw the text.

#### 26.62.3.4 `t_max_err jtextlayout_getchar (t_jtextlayout * tl, long index, long * pch)`

Retrieve the unicode character at a given index.

**Parameters:**

*tl* The textlayout object to query.

*index* The index from which to fetch the unicode character.

*pch* The address of a variable to hold the unicode character value upon return.

**Returns:**

A Max error code.

#### 26.62.3.5 `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:**

*tl* The textlayout object to query.

*index* The index from which to fetch the unicode character.

*rect* The address of a valid [t\\_rect](#) which will be filled in upon return.

**Returns:**

A Max error code.

#### 26.62.3.6 `long jtextlayout_getnumchars (t_jtextlayout * tl)`

Retrieve a count of the number of characters in a textlayout object.

**Parameters:**

*tl* The textlayout object to query.

**Returns:**

The number of characters.

#### 26.62.3.7 `void jtextlayout_measure (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:**

*textlayout* The textlayout object to query.

*startindex* You can measure a subset of the characters. This defines the character from which to start.

*numchars* Pass -1 for all characters from startindex to end

*includewhitespace* Define whether to measure with or without whitespace truncated from edges.

*width* Returns the width of text not including any margins.

*height* Returns the height of text not including any margins.

*numlines* Returns the number of lines of text.

**26.62.3.8** 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:**

*textlayout* The textlayout object.  
*utf8* The text to render.  
*jfont* The font with which to render the text.  
*x* The text is placed within rect specified by x, y, width, height.  
*y* The text is placed within rect specified by x, y, width, height.  
*width* The text is placed within rect specified by x, y, width, height.  
*height* The text is placed within rect specified by x, y, width, height.  
*justification* How to justify the text within the rect.  
*flags* Additional flags to control behaviour.

**26.62.3.9** void jtextlayout\_settextcolor (t\_jtextlayout \* *textlayout*, t\_jrgba \* *textcolor*)

Set the color to render text in a textlayout object.

**Parameters:**

*textlayout* The textlayout object for which to set the color.  
*textcolor* The color for the text.

**26.62.3.10** 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:**

*g* The graphics context for the textlayout.  
*bgcolor* The background color for the textlayout.

**Returns:**

The new textlayout object.

## 26.63 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 jpopupmenu_setfont (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.*

### 26.63.1 Detailed Description

Popup menu API so externals can create popup menus that can also be drawn into.

### 26.63.2 Function Documentation

#### 26.63.2.1 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:

- menu* The pop-up menu to which the item will be added.
- itemid* Each menu item should be assigned a unique integer id using this parameter.
- utf8Text* The text to display in for the menu item.
- textColor* The color to use for the menu item, or NULL to use the default color.
- checked* A non-zero value indicates that the item should have a check-mark next to it.
- disabled* A non-zero value indicates that the item should be disabled.
- icon* A [t\\_jsurface](#) will be used as an icon for the menu item if provided here. Pass NULL for no icon.

#### 26.63.2.2 void jpopupmenu\_addseparator (t\_jpopupmenu \* menu)

Add a separator to a pop-menu.

##### Parameters:

- menu* The pop-up menu to which the separator will be added.

#### 26.63.2.3 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.

##### Parameters:

- menu* The pop-up menu to which a menu will be added as a submenu.
- utf8Name* The name of the menu item.
- submenu* The pop-up menu which will be used as the submenu.
- disabled* Pass a non-zero value to disable the menu item.

#### 26.63.2.4 void jpopupmenu\_clear (t\_jpopupmenu \* menu)

Clear the contents of a pop-up menu.

##### Parameters:

- menu* The pop-up menu whose contents will be cleared.

### 26.63.2.5 `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.

### 26.63.2.6 `void jpopupmenu_destroy (t_jpopupmenu * menu)`

Free a pop-up menu created with [jpopupmenu\\_create\(\)](#).

#### Parameters:

*menu* The pop-up menu to be freed.

### 26.63.2.7 `int jpopupmenu_popup (t_jpopupmenu * menu, t_pt screen, int defitemid)`

Tell a menu to display at a specified location.

#### Parameters:

*menu* The pop-up menu to display.

*screen* The point at which to display in screen coordinates.

*defitemid* The initially choosen item id.

#### Returns:

The item id for the item in the menu choosen by the user.

### 26.63.2.8 `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:

*menu* The pop-up menu to display.

*box* The box above which to display the menu.

*view* The patcherview for the box in which to display the menu.

*offset* An offset from the box position at which to display the menu.

*defitemid* The initially choosen item id.

#### Returns:

The item id for the item in the menu choosen by the user.



**26.63.2.9** `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:**

*menu* The pop-up menu to display.

*box* The box above which to display the menu.

*view* The patcherview for the box in which to display the menu.

*defitemid* The initially choosen item id.

**Returns:**

The item id for the item in the menu choosen by the user.

**26.63.2.10** `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:**

*menu* The pop-up menu to which the colors will be applied.

*text* The text color for menu items.

*bg* The background color for menu items.

*highlightedtext* The text color for the highlighted menu item.

*highlightedbg* The background color the highlighted menu item.

**26.63.2.11** `void jpopupmenu_setfont (t_jpopupmenu * menu, t_jfont * font)`

Set the font used by a pop-up menu.

**Parameters:**

*menu* The pop-up menu whose font will be set.

*font* A pointer to a font object, whose font info will be copied to the pop-up menu.

## 26.64 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.*

### 26.64.1 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 t
ime
    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.

## 26.64.2 Function Documentation

### 26.64.2.1 `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:**

- b* The object/box for the layer opened by [jbox\\_start\\_layer\(\)](#).
- view* The patcherview for the object opened by [jbox\\_start\\_layer\(\)](#).
- name* The name of the layer.

**Returns:**

A Max error code.

### 26.64.2.2 `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:**

- b* The object/box to invalidate.
- view* The patcherview for the object which should be invalidated, or NULL for all patcherviews.
- name* The name of the layer to invalidate.

**Returns:**

A Max error code.

### 26.64.2.3 `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:**

- b* The object/box to be painted.
- view* The patcherview for the object which should be painted, or NULL for all patcherviews.
- name* The name of the layer to paint.
- x* The x-coordinate for the position at which to paint the layer.
- y* The y-coordinate for the position at which to paint the layer.

**Returns:**

A Max error code.

#### 26.64.2.4 `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:

- b* The object/box to which the layer is attached.
- view* The patcherview for the object to which the layer is attached.
- name* A name for this layer.
- width* The width of the layer.
- height* The height of the layer.

##### Returns:

- A `t_jgraphics` context for drawing into the layer.

## 26.65 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.*

#### 26.65.1 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.

## 26.65.2 Function Documentation

### 26.65.2.1 `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:**

*dv* The dataview instance.

**Returns:**

A pointer to the dataview's client object.

### 26.65.2.2 `void* jdataview_new (void)`

Create a dataview. You should free it with [object\\_free\(\)](#).

**Returns:**

A pointer to the new instance.

### 26.65.2.3 `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:**

*dv* The dataview instance.

*client* The object to be assigned as the dataview's client.

## 26.66 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\\_unicotetoutf8](#) (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.*

#### 26.66.1 Detailed Description

#### 26.66.2 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;`

### 26.66.2.1 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, &cvtbu
        ffer, &cvtbuflen) == ERR_NONE) {
        // do something with the converted buffer
        system_freeptr(cvtbuffer); // free newly allocated data buffer
    }
    object_free(conv); // free converter
}
```

### 26.66.3 Function Documentation

#### 26.66.3.1 **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:

- src\_encoding* The name encoding of the input.
- in* The input string.
- inbytes* The number of bytes in the input string.
- dest\_encoding* The name of the encoding to use for the output.
- out* The address of a char\*, which will be allocated and filled with the string in the new encoding.
- outbytes* 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 `system_freeptr(*out)` to free any allocated memory.

#### 26.66.3.2 **char\*** `charset_unicotetoutf8 (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:

- s* An array of wide (16-bit) unicode characters.
- len* The length of s.
- outlen* 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.

**26.66.3.3 long charset\_utf8\_count (char \* *utf8*, long \* *bytecount*)**

Returns utf8 character count, and optionally bytecount.

**Parameters:**

*utf8* The UTF-8 encoded string whose characters are to be counted.

*bytecount* 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.

**26.66.3.4 char\* charset\_utf8\_offset (char \* *utf8*, long *charoffset*, long \* *byteoffset*)**

Returns utf8 character offset (positive or negative), and optionally byte offset.

**Parameters:**

*utf8* A UTF-8 encoded string.

*charoffset* The char offset into the string at which to find the byte offset.

*byteoffset* 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.

**26.66.3.5 unsigned short\* charset\_utf8tounicode (char \* *s*, long \* *outlen*)**

Convert a UTF8 C-String into a 16-bit-wide-character array.

**Parameters:**

*s* The string to be converted to unicode.

*outlen* 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.



## Chapter 27

# Data Structure Documentation

### 27.1 Ex\_ex Struct Reference

ex\_ex.

```
#include <ext_expr.h>
```

#### Data Fields

- union {  
    } [ex\\_cont](#)  
  
    *content*
- long [ex\\_type](#)  
    *type of the node*

#### 27.1.1 Detailed Description

ex\_ex.

Definition at line 43 of file ext\_expr.h.

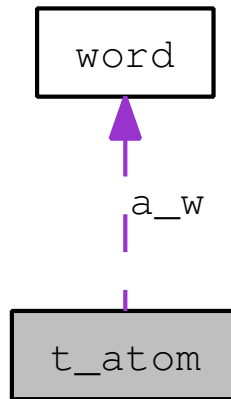
The documentation for this struct was generated from the following file:

- ext\_expr.h

## 27.2 t\_atom Struct Reference

An atom is a typed datum.

`#include <ext_mess.h>` Collaboration diagram for `t_atom`:



### Data Fields

- short `a_type`  
*a value as defined in `e_max_atomtypes`*
- union `word a_w`  
*the actual data*

### 27.2.1 Detailed Description

An atom is a typed datum.

Definition at line 246 of file `ext_mess.h`.

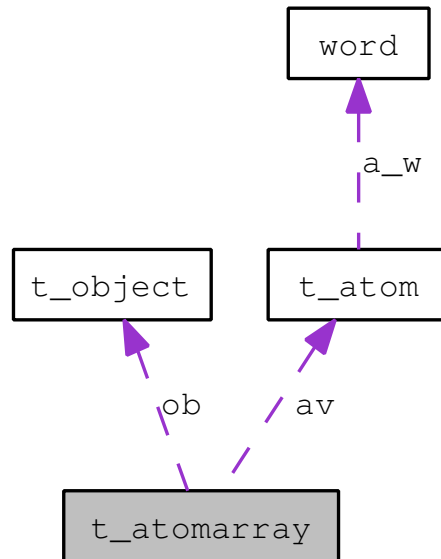
The documentation for this struct was generated from the following file:

- `ext_mess.h`

## 27.3 t\_atomarray Struct Reference

The atomarray object.

`#include <ext_atomarray.h>` Collaboration diagram for t\_atomarray:



### 27.3.1 Detailed Description

The atomarray object. This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

Definition at line 17 of file `ext_atomarray.h`.

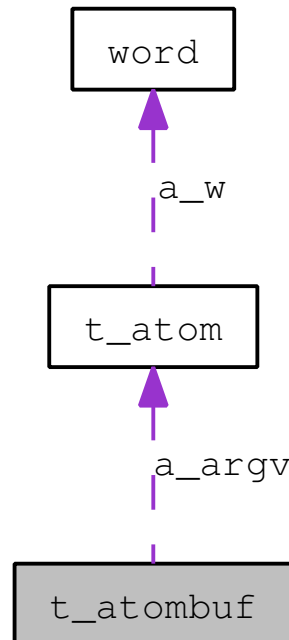
The documentation for this struct was generated from the following file:

- `ext_atomarray.h`

## 27.4 t\_atombuf Struct Reference

The atombuf struct provides a way to pass a collection of atoms.

`#include <ext_maxtypes.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*

#### 27.4.1 Detailed Description

The atombuf struct provides a way to pass a collection of atoms.

Definition at line 58 of file `ext_maxtypes.h`.

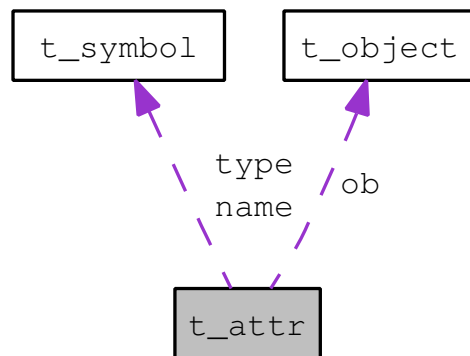
The documentation for this struct was generated from the following file:

- `ext_maxtypes.h`

## 27.5 t\_attr Struct Reference

Common attr struct.

`#include <ext_obex.h>` Collaboration diagram for t\_attr:



### 27.5.1 Detailed Description

Common attr struct. This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

Definition at line 141 of file `ext_obex.h`.

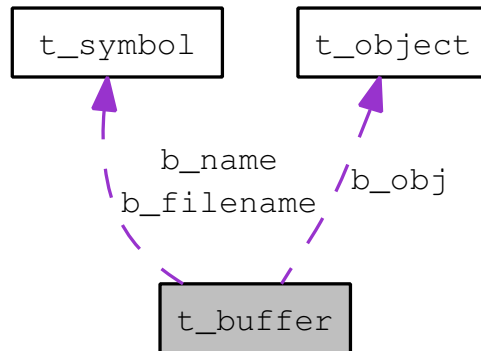
The documentation for this struct was generated from the following file:

- `ext_obex.h`

## 27.6 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\\_loversr](#)  
*1 / sr*
- float [b\\_msr](#)  
*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 readagain message*
- [long b\\_oldnchans](#)  
*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)*
- [Boolean 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*

### 27.6.1 Detailed Description

Data structure for the `buffer~` object.

Definition at line 29 of file `buffer.h`.

The documentation for this struct was generated from the following file:

- `buffer.h`

## 27.7 t\_celldesc Struct Reference

A dataview cell description.

```
#include <jdataview.h>
```

### 27.7.1 Detailed Description

A dataview cell description.

Definition at line 106 of file jdataview.h.

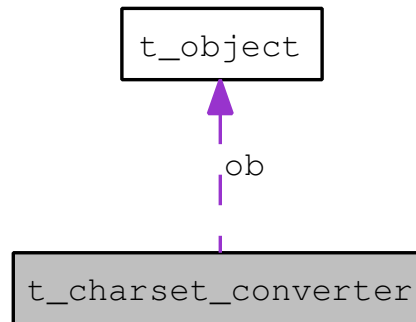
The documentation for this struct was generated from the following files:

- jdataview.h
- [scripto.c](#)

## 27.8 t\_charset\_converter Struct Reference

The charset\_converter object.

`#include <ext_charset.h>` Collaboration diagram for t\_charset\_converter:



### 27.8.1 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.

Definition at line 28 of file `ext_charset.h`.

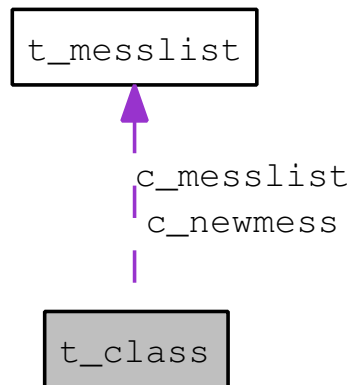
The documentation for this struct was generated from the following file:

- `ext_charset.h`

## 27.9 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*

### 27.9.1 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.

Definition at line 153 of file `ext_mess.h`.

The documentation for this struct was generated from the following file:

- `ext_mess.h`

## 27.10 t\_datetime Struct Reference

The Systime data structure.

```
#include <ext_systime.h>
```

### Data Fields

- unsigned long [year](#)  
*year*
- unsigned long [month](#)  
*month*
- unsigned long [day](#)  
*day*
- unsigned long [hour](#)  
*hour*
- unsigned long [minute](#)  
*minute*
- unsigned long [second](#)  
*second*
- unsigned long [millisecond](#)  
*(reserved for future use)*

### 27.10.1 Detailed Description

The Systime data structure.

Definition at line 20 of file ext\_systime.h.

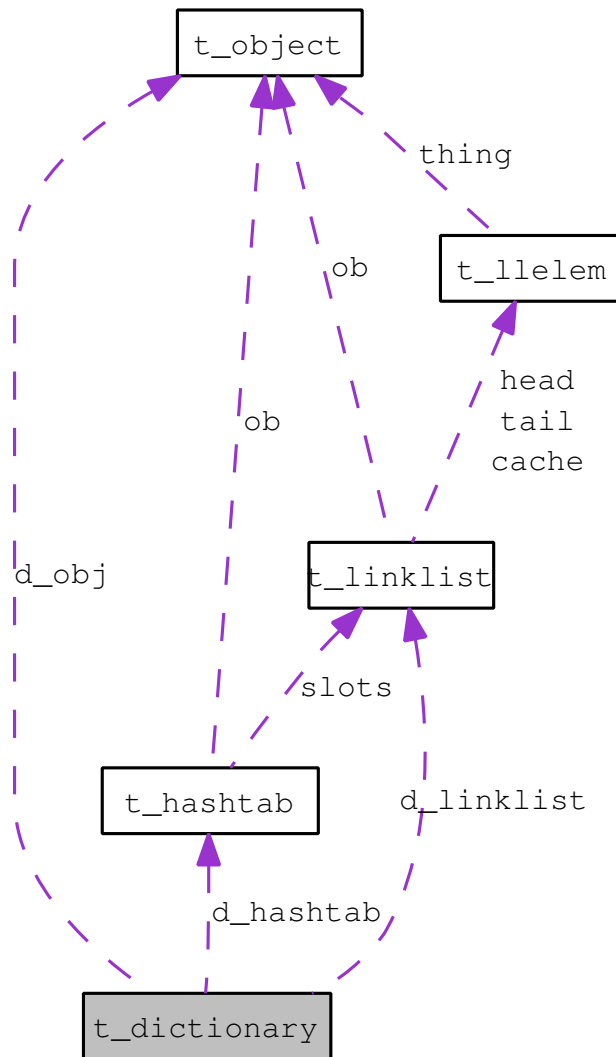
The documentation for this struct was generated from the following file:

- ext\_systime.h

## 27.11 t\_dictionary Struct Reference

The dictionary object.

`#include <ext_dictionary.h>` Collaboration diagram for t\_dictionary:



### 27.11.1 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](#)

Definition at line 42 of file `ext_dictionary.h`.

The documentation for this struct was generated from the following file:

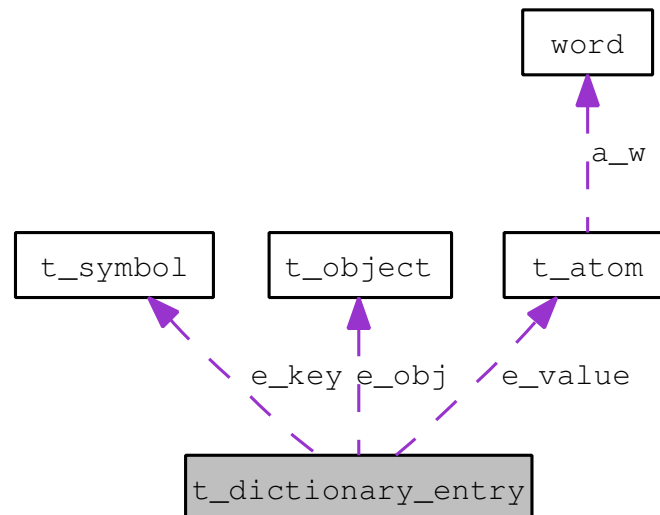
- `ext_dictionary.h`



## 27.12 t\_dictionary\_entry Struct Reference

A dictionary entry.

`#include <ext_dictionary.h>` Collaboration diagram for t\_dictionary\_entry:



### 27.12.1 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](#)

Definition at line 29 of file `ext_dictionary.h`.

The documentation for this struct was generated from the following file:

- `ext_dictionary.h`

## 27.13 t\_expr Struct Reference

Struct for an instance of expr.

```
#include <ext_expr.h>
```

### Data Fields

- struct ex\_ex [exp\\_res](#)  
*the result of last evaluation*

### 27.13.1 Detailed Description

Struct for an instance of expr.

Definition at line 81 of file ext\_expr.h.

The documentation for this struct was generated from the following file:

- ext\_expr.h

## 27.14 t\_fileinfo Struct Reference

Information about a file.

```
#include <ext_path.h>
```

### Data Fields

- long [type](#)  
*type (four-char-code)*
- long [creator](#)  
*Mac-only creator (four-char-code).*
- long [date](#)  
*date*
- long [flags](#)  
*One of the values defined in [e\\_max\\_fileinfo\\_flags](#).*

### 27.14.1 Detailed Description

Information about a file.

Definition at line 153 of file `ext_path.h`.

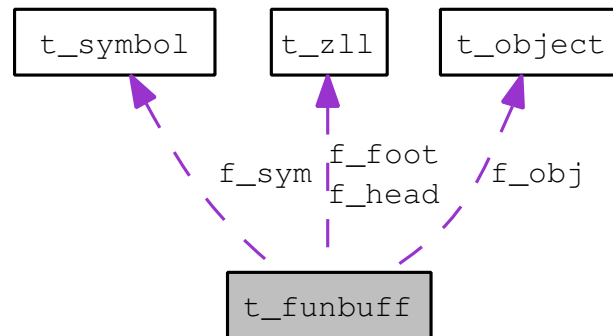
The documentation for this struct was generated from the following file:

- `ext_path.h`

## 27.15 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*

### 27.15.1 Detailed Description

The structure of a funbuff object.

Definition at line 79 of file ext\_maxtypes.h.

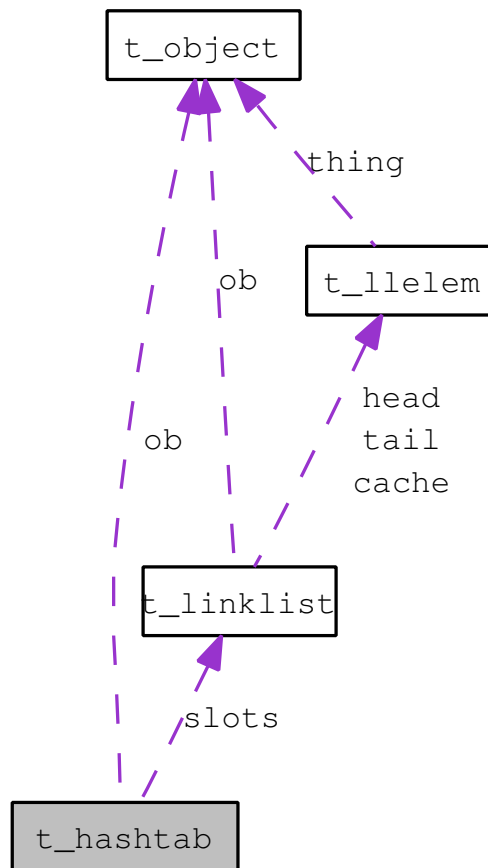
The documentation for this struct was generated from the following file:

- ext\_maxtypes.h

## 27.16 t\_hashtab Struct Reference

The hashtab object.

`#include <ext_hashtab.h>` Collaboration diagram for `t_hashtab`:



### 27.16.1 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](#)

Definition at line 40 of file `ext_hashtab.h`.

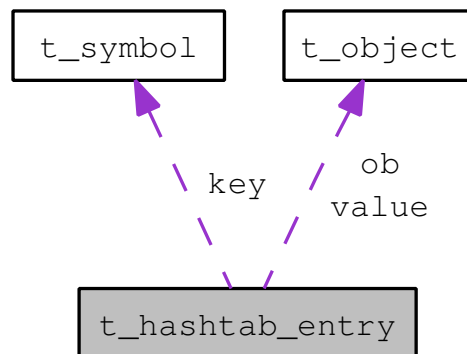
The documentation for this struct was generated from the following file:

- `ext_hashtab.h`

## 27.17 t\_hashtab\_entry Struct Reference

A hashtab entry.

#include <ext\_hashtab.h> Collaboration diagram for t\_hashtab\_entry:



### 27.17.1 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](#)

Definition at line 26 of file `ext_hashtab.h`.

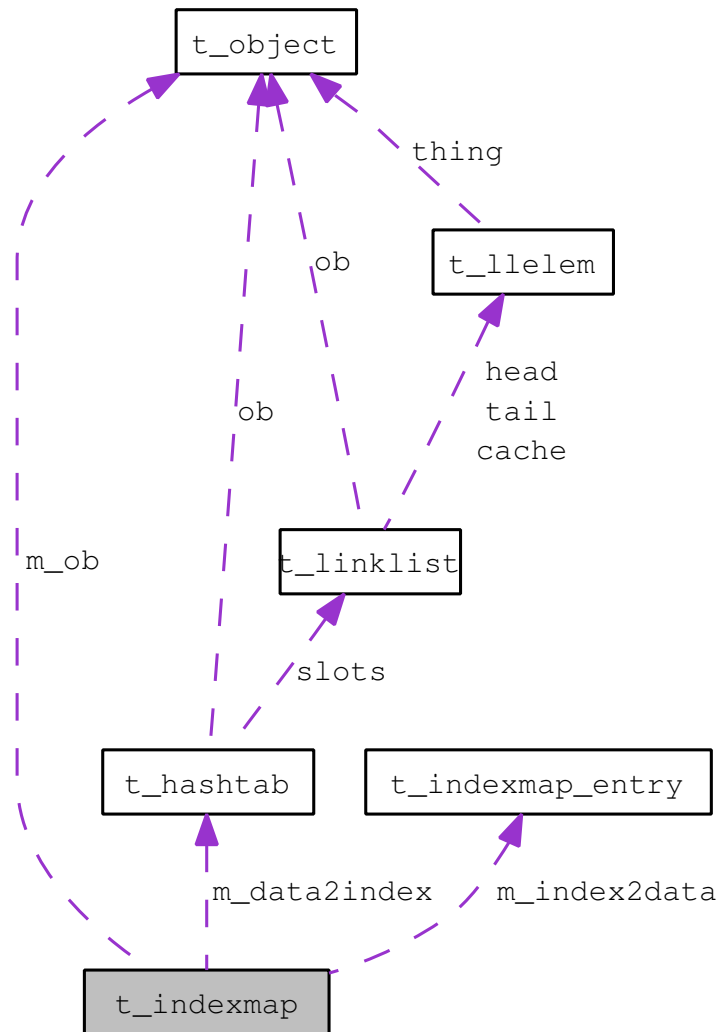
The documentation for this struct was generated from the following file:

- `ext_hashtab.h`

## 27.18 t\_indexmap Struct Reference

An indexmap object.

#include <indexmap.h> Collaboration diagram for t\_indexmap:



### 27.18.1 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](#)

Definition at line 32 of file `indexmap.h`.

The documentation for this struct was generated from the following file:



- indexmap.h

## 27.19 t\_indexmap\_entry Struct Reference

An indexmap element.

```
#include <indexmap.h>
```

### 27.19.1 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](#)

Definition at line 18 of file indexmap.h.

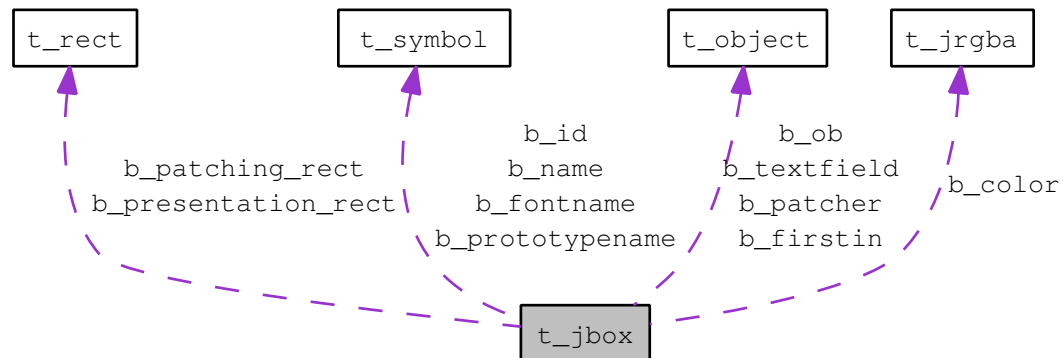
The documentation for this struct was generated from the following file:

- indexmap.h

## 27.20 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:



### 27.20.1 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 it's members directly any code.

Definition at line 126 of file `jpatcher_api.h`.

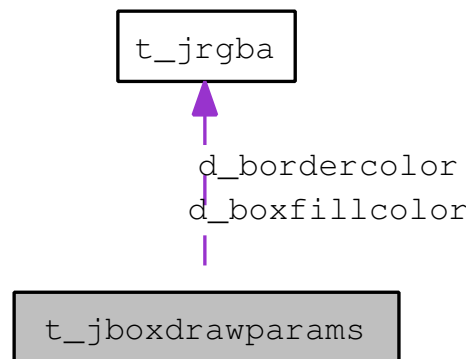
The documentation for this struct was generated from the following file:

- `jpatcher_api.h`

## 27.21 t\_jboxdrawparams Struct Reference

The [t\\_jboxdrawparams](#) structure.

#include <jpatcher\_api.h> Collaboration diagram for t\_jboxdrawparams:



### 27.21.1 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.

Definition at line 104 of file `jpatcher_api.h`.

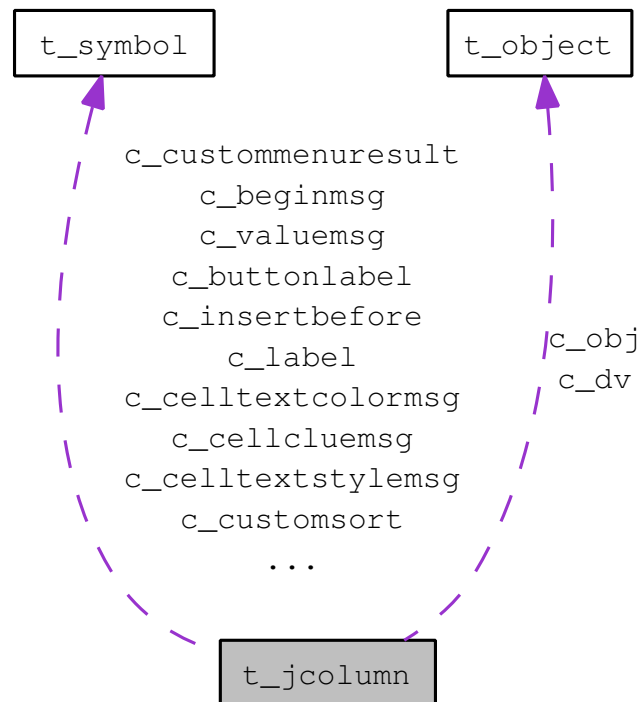
The documentation for this struct was generated from the following file:

- `jpatcher_api.h`

## 27.22 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_fontsize`  
*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\_FORWARDS 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)*
- 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*

### 27.22.1 Detailed Description

A dataview column. Columns for a given dataview are stored in a [t\\_hashtab](#) and accessed by name.

Definition at line 117 of file `jdataview.h`.

The documentation for this struct was generated from the following file:

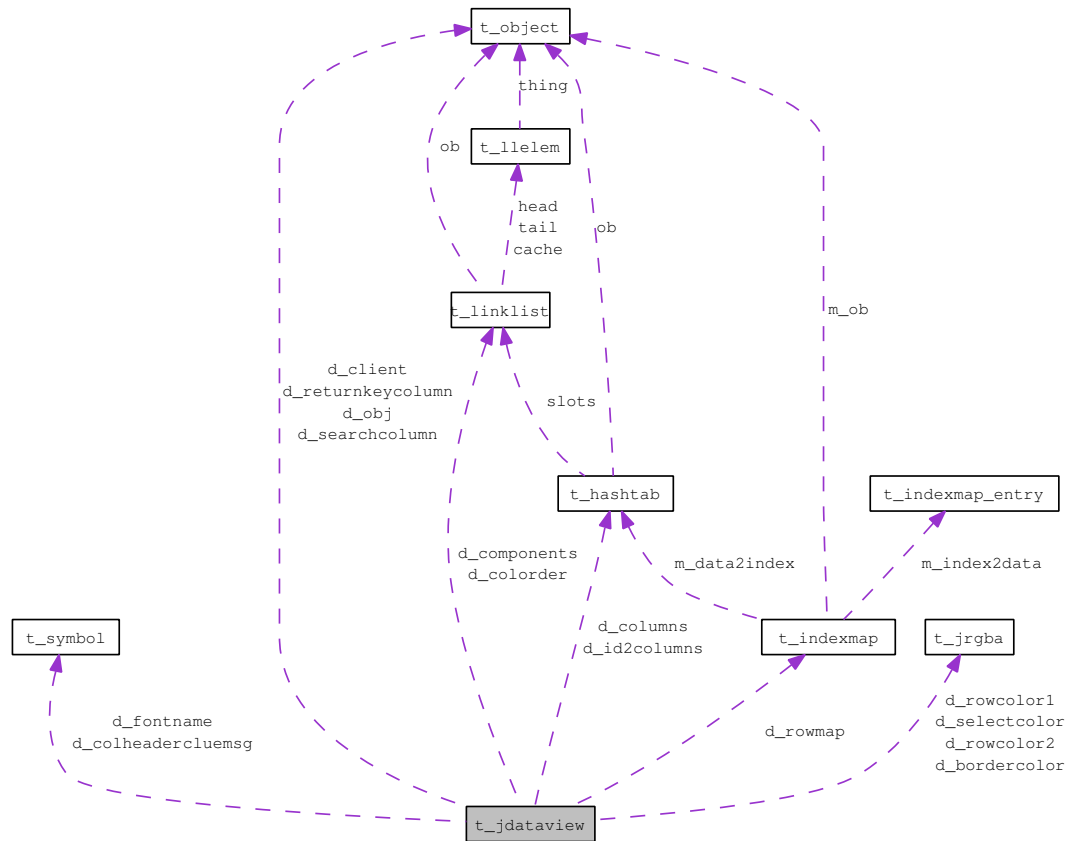
- `jdataview.h`



## 27.23 t\_jdataview Struct Reference

The dataview object.

#include <jdataview.h> Collaboration diagram for t\_jdataview:



### 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_colorder`  
current order of columns

- [t\\_indexmap \\* d\\_rowmap](#)  
*collection of rows (including number of rows)*
- 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\\_cansselectmultiple](#)  
*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\\_enableddeletekey](#)  
*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)*

### 27.23.1 Detailed Description

The dataview object.

Definition at line 174 of file `jdataview.h`.

The documentation for this struct was generated from the following file:

- `jdataview.h`

## 27.24 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.*

### 27.24.1 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.

Definition at line 722 of file jgraphics.h.

The documentation for this struct was generated from the following file:

- jgraphics.h

## 27.25 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*

### 27.25.1 Detailed Description

An affine transformation (such as scale, shear, etc).

Definition at line 1002 of file jgraphics.h.

The documentation for this struct was generated from the following file:

- jgraphics.h

## 27.26 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].*

### 27.26.1 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](#)

Definition at line 81 of file jpatcher\_api.h.

The documentation for this struct was generated from the following file:

- jpatcher\_api.h

## 27.27 t\_jrgba 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].*

### 27.27.1 Detailed Description

A color composed of red, green, blue, and alpha components.

Definition at line 90 of file jpatcher\_api.h.

The documentation for this struct was generated from the following file:

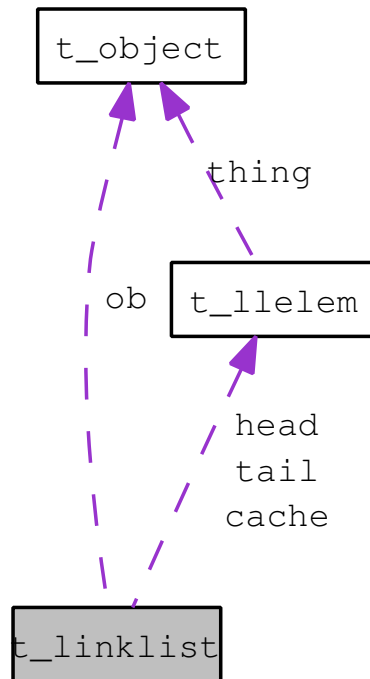
- jpatcher\_api.h



## 27.28 t\_linklist Struct Reference

The linklist object.

`#include <ext_linklist.h>` Collaboration diagram for t\_linklist:



### 27.28.1 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](#)

Definition at line 31 of file `ext_linklist.h`.

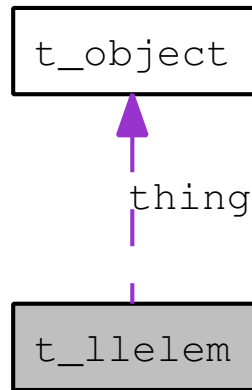
The documentation for this struct was generated from the following file:

- `ext_linklist.h`

## 27.29 t\_llelem Struct Reference

A linklist element.

`#include <ext_linklist.h>` Collaboration diagram for `t_llelem`:



### 27.29.1 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](#)

Definition at line 17 of file `ext_linklist.h`.

The documentation for this struct was generated from the following file:

- `ext_linklist.h`

## 27.30 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](#) [MAXARG+1]  
*Argument type information.*

### 27.30.1 Detailed Description

A list of symbols and their corresponding methods, complete with typechecking information.

Definition at line 108 of file ext\_mess.h.

The documentation for this struct was generated from the following file:

- ext\_mess.h

## 27.31 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.*
- long [o\\_magic](#)  
*magic number*
- struct inlet \* [o\\_inlet](#)  
*list of inlets*
- struct outlet \* [o\\_outlet](#)  
*list of outlets*

### 27.31.1 Detailed Description

The structure for the head of any object which wants to have inlets or outlets, or support attributes.

Definition at line 135 of file ext\_mess.h.

The documentation for this struct was generated from the following file:

- ext\_mess.h

## 27.32 t\_path Struct Reference

The path data structure.

```
#include <ext_path.h>
```

### 27.32.1 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.

Definition at line 165 of file ext\_path.h.

The documentation for this struct was generated from the following file:

- ext\_path.h

## 27.33 t\_pathlink Struct Reference

The pathlink data structure.

```
#include <ext_path.h>
```

### 27.33.1 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.

Definition at line 177 of file ext\_path.h.

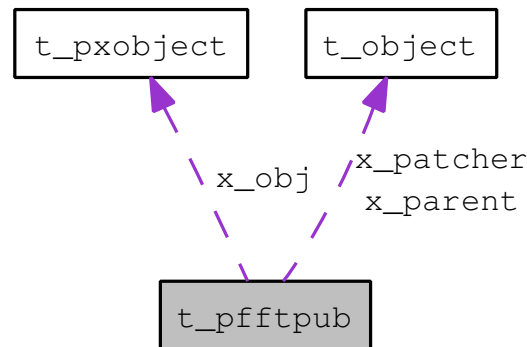
The documentation for this struct was generated from the following file:

- ext\_path.h

## 27.34 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_patcher`  
*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_fftoffset`  
*n samples offset before fft is started*
- `long x_fftindex`  
*current index into fft frame*
- `short x_fullspect`  
*process half-spectrum (0) or full mirrored spectrum (1)?*

### 27.34.1 Detailed Description

Public FFT Patcher struct.

Definition at line 12 of file r\_pfft.h.

The documentation for this struct was generated from the following file:

- r\_pfft.h





### 27.35.1 Detailed Description

used to pass data to a client sort function

Definition at line 242 of file jdataview.h.

The documentation for this struct was generated from the following file:

- jdataview.h

## 27.36 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.*

### 27.36.1 Detailed Description

Coordinates for specifying a point.

**See also:**

[t\\_rect](#)  
[t\\_size](#)

Definition at line 58 of file `jpatcher_api.h`.

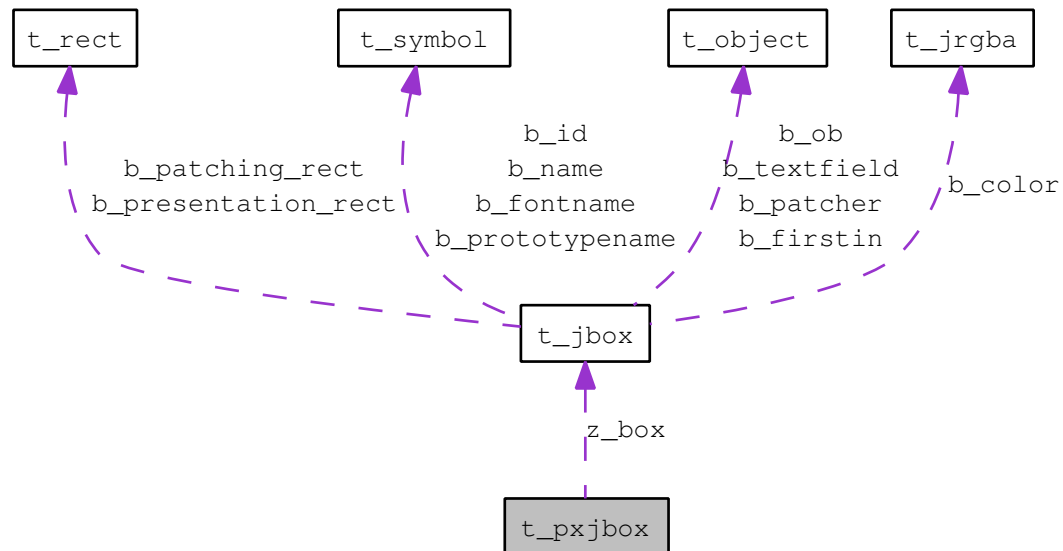
The documentation for this struct was generated from the following file:

- `jpatcher_api.h`

## 27.37 t\_pxjbox Struct Reference

Header for any ui signal processing object.

`#include <z_dsp.h>` Collaboration diagram for `t_pxjbox`:



### Data Fields

- `t_jbox 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`  
*an array that indicates what inlets/outlets are connected with signals*
- short `z_misc`  
*flags (bitmask) determining object behaviour, such as `Z_NO_INPLACE`, `Z_PUT_FIRST`, or `Z_PUT_LAST`*

### 27.37.1 Detailed Description

Header for any ui signal processing object. For non-ui objects use `t_pxobject`.

Definition at line 411 of file `z_dsp.h`.

The documentation for this struct was generated from the following file:

- `z_dsp.h`

## 27.38 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](#)  
*an array that indicates what inlets/outlets are connected with signals*
- short [z\\_misc](#)  
*flags (bitmask) determining object behaviour, such as [Z\\_NO\\_INPLACE](#), [Z\\_PUT\\_FIRST](#), or [Z\\_PUT\\_LAST](#)*

### 27.38.1 Detailed Description

Header for any non-ui signal processing object. For ui objects use [t\\_pxjbox](#).

Definition at line 88 of file `z_dsp.h`.

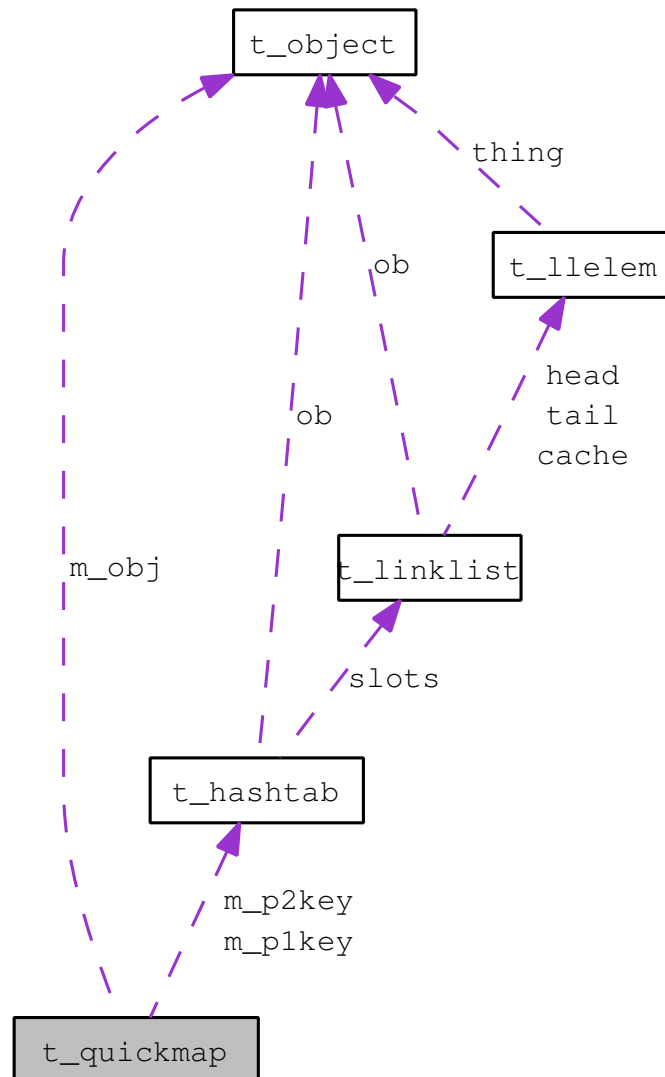
The documentation for this struct was generated from the following file:

- `z_dsp.h`

## 27.39 t\_quickmap Struct Reference

The quickmap object.

#include <ext\_quickmap.h> Collaboration diagram for t\_quickmap:



### 27.39.1 Detailed Description

The quickmap object. This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

Definition at line 17 of file `ext_quickmap.h`.

The documentation for this struct was generated from the following file:

- `ext_quickmap.h`

## 27.40 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.*

### 27.40.1 Detailed Description

Coordinates for specifying a rectangular region.

**See also:**

[t\\_pt](#)  
[t\\_size](#)

Definition at line 44 of file `jpatcher_api.h`.

The documentation for this struct was generated from the following file:

- `jpatcher_api.h`

## 27.41 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](#)  
*An array of buffers holding the vectors of audio.*
- float [s\\_sr](#)  
*The sample rate of the signal.*

### 27.41.1 Detailed Description

The signal data structure.

Definition at line 106 of file `z_dsp.h`.

The documentation for this struct was generated from the following file:

- `z_dsp.h`



## 27.42 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.*

### 27.42.1 Detailed Description

Coordinates for specifying the size of a region.

**See also:**

[t\\_rect](#)

[t\\_pt](#)

Definition at line 70 of file `jpatcher_api.h`.

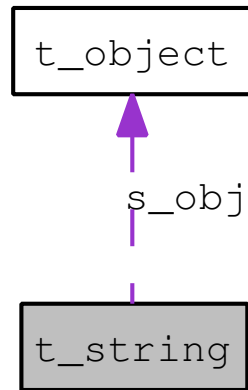
The documentation for this struct was generated from the following file:

- `jpatcher_api.h`

## 27.43 t\_string Struct Reference

The string object.

`#include <ext_obstring.h>` Collaboration diagram for t\_string:



### 27.43.1 Detailed Description

The string object. This struct is provided for debugging convenience, but should be considered opaque and is subject to change without notice.

Definition at line 28 of file `ext_obstring.h`.

The documentation for this struct was generated from the following file:

- `ext_obstring.h`

## 27.44 t\_symbol Struct Reference

The symbol.

```
#include <ext_mess.h>
```

### Data Fields

- char \* [s\\_name](#)  
*name: a c-string*
- struct object \* [s\\_thing](#)  
*possible binding to a [t\\_object](#)*

#### 27.44.1 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.

Definition at line 72 of file ext\_mess.h.

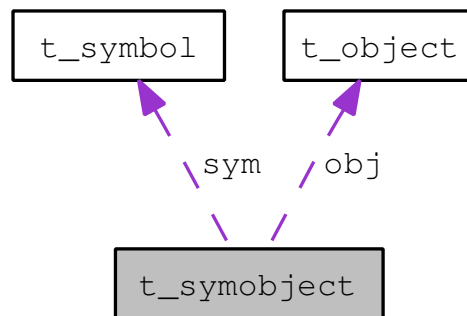
The documentation for this struct was generated from the following file:

- ext\_mess.h

## 27.45 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.*

### 27.45.1 Detailed Description

The symobject data structure.

Definition at line 14 of file `ext_symobject.h`.

The documentation for this struct was generated from the following file:

- `ext_symobject.h`

## 27.46 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*

### 27.46.1 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.

Definition at line 121 of file `ext_mess.h`.

The documentation for this struct was generated from the following file:

- `ext_mess.h`

## 27.47 t\_zll Struct Reference

A simple doubly-linked list used by the [t\\_funbuff](#) object.

```
#include <ext_maxtypes.h>
```

### 27.47.1 Detailed Description

A simple doubly-linked list used by the [t\\_funbuff](#) object.

Definition at line 67 of file ext\_maxtypes.h.

The documentation for this struct was generated from the following file:

- ext\_maxtypes.h

## 27.48 word Union Reference

Union for packing any of the datum defined in [e\\_max\\_atomtypes](#).

```
#include <ext_mess.h>
```

### Data Fields

- long [w\\_long](#)  
*long integer*
- 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*

### 27.48.1 Detailed Description

Union for packing any of the datum defined in [e\\_max\\_atomtypes](#).

Definition at line 234 of file `ext_mess.h`.

The documentation for this union was generated from the following file:

- `ext_mess.h`

# Index

A\_CANT  
  atom, [299](#)  
A\_COMMA  
  atom, [299](#)  
A\_DEFER  
  atom, [299](#)  
A\_DEFER\_LOW  
  atom, [299](#)  
A\_DEFFLOAT  
  atom, [299](#)  
A\_DEFLONG  
  atom, [299](#)  
A\_DEFSYM  
  atom, [299](#)  
A\_DOLLAR  
  atom, [299](#)  
A\_DOLLSYM  
  atom, [299](#)  
A\_FLOAT  
  atom, [299](#)  
A\_GIMME  
  atom, [299](#)  
A\_GIMMEBACK  
  atom, [299](#)  
A\_LONG  
  atom, [299](#)  
A\_NOTHING  
  atom, [299](#)  
A\_OBJ  
  atom, [299](#)  
A\_SEMI  
  atom, [299](#)  
A\_SYM  
  atom, [299](#)  
A\_USURP  
  atom, [299](#)  
A\_USURP\_LOW  
  atom, [300](#)  
aaCancel  
  misc, [360](#)  
aaNo  
  misc, [360](#)  
aaYes  
  misc, [360](#)  
addbang  
  class\_old, [198](#)  
addfloat  
  class\_old, [198](#)  
addftx  
  class\_old, [198](#)  
addint  
  class\_old, [198](#)  
addinx  
  class\_old, [198](#)  
address  
  class\_old, [199](#)  
alias  
  class\_old, [199](#)  
atom  
  A\_CANT, [299](#)  
  A\_COMMA, [299](#)  
  A\_DEFER, [299](#)  
  A\_DEFER\_LOW, [299](#)  
  A\_DEFFLOAT, [299](#)  
  A\_DEFLONG, [299](#)  
  A\_DEFSYM, [299](#)  
  A\_DOLLAR, [299](#)  
  A\_DOLLSYM, [299](#)  
  A\_FLOAT, [299](#)  
  A\_GIMME, [299](#)  
  A\_GIMMEBACK, [299](#)  
  A\_LONG, [299](#)  
  A\_NOTHING, [299](#)  
  A\_OBJ, [299](#)  
  A\_SEMI, [299](#)  
  A\_SYM, [299](#)  
  A\_USURP, [299](#)  
  A\_USURP\_LOW, [300](#)  
  atom\_alloc, [300](#)  
  atom\_alloc\_array, [300](#)  
  atom\_arg\_getdouble, [300](#)  
  atom\_arg\_getfloat, [301](#)  
  atom\_arg\_getlong, [301](#)  
  atom\_arg\_getobjclass, [302](#)  
  atom\_arg\_getsym, [302](#)  
  atom\_copy, [303](#)  
  atom\_getatom\_array, [303](#)  
  atom\_getchar\_array, [303](#)  
  atom\_getcharfix, [304](#)  
  atom\_getdouble\_array, [304](#)



- atom\_getfloat, 304
- atom\_getfloat\_array, 305
- atom\_getformat, 305
- atom\_getlong, 305
- atom\_getlong\_array, 306
- atom\_getobj, 306
- atom\_getobj\_array, 306
- atom\_getobjclass, 307
- atom\_getsym, 307
- atom\_getsym\_array, 307
- atom\_gettext, 307
- atom\_gettype, 308
- atom\_setatom\_array, 308
- atom\_setchar\_array, 308
- atom\_setdouble\_array, 309
- atom\_setfloat, 309
- atom\_setfloat\_array, 309
- atom\_setformat, 310
- atom\_setlong, 310
- atom\_setlong\_array, 311
- atom\_setobj, 311
- atom\_setobj\_array, 311
- atom\_setparse, 311
- atom\_setsym, 312
- atom\_setsym\_array, 312
- atomisatomarray, 312
- atomisdictionary, 313
- atomisstring, 313
- e\_max\_atom\_gettext\_flags, 298
- e\_max\_atomtypes, 299
- OBEX\_UTIL\_ATOM\_GETTEXT\_-  
COMMA\_DELIM, 299
- OBEX\_UTIL\_ATOM\_GETTEXT\_-  
DEFAULT, 298
- OBEX\_UTIL\_ATOM\_GETTEXT\_FORCE\_-  
ZEROS, 299
- OBEX\_UTIL\_ATOM\_GETTEXT\_NUM\_-  
HI\_RES, 299
- OBEX\_UTIL\_ATOM\_GETTEXT\_SYM\_-  
FORCE\_QUOTE, 299
- OBEX\_UTIL\_ATOM\_GETTEXT\_SYM\_-  
NO\_QUOTE, 299
- OBEX\_UTIL\_ATOM\_GETTEXT\_-  
TRUNCATE\_ZEROS, 298
- postargs, 313
- Atom Array, 213
- atom\_alloc
  - atom, 300
- atom\_alloc\_array
  - atom, 300
- atom\_arg\_getdouble
  - atom, 300
- atom\_arg\_getfloat
  - atom, 301
- atom\_arg\_getlong
  - atom, 301
- atom\_arg\_getobjclass
  - atom, 302
- atom\_arg\_getsym
  - atom, 302
- atom\_copy
  - atom, 303
- atom\_getatom\_array
  - atom, 303
- atom\_getchar\_array
  - atom, 303
- atom\_getcharfix
  - atom, 304
- atom\_getdouble\_array
  - atom, 304
- atom\_getfloat
  - atom, 304
- atom\_getfloat\_array
  - atom, 305
- atom\_getformat
  - atom, 305
- atom\_getlong
  - atom, 305
- atom\_getlong\_array
  - atom, 306
- atom\_getobj
  - atom, 306
- atom\_getobj\_array
  - atom, 306
- atom\_getobjclass
  - atom, 307
- atom\_getsym
  - atom, 307
- atom\_getsym\_array
  - atom, 307
- atom\_gettext
  - atom, 307
- atom\_gettype
  - atom, 308
- atom\_setatom\_array
  - atom, 308
- atom\_setchar\_array
  - atom, 308
- atom\_setdouble\_array
  - atom, 309
- atom\_setfloat
  - atom, 309
- atom\_setfloat\_array
  - atom, 309
- atom\_setformat
  - atom, 310
- atom\_setlong
  - atom, 310

- atom\_setlong\_array
  - atom, [311](#)
- atom\_setobj
  - atom, [311](#)
- atom\_setobj\_array
  - atom, [311](#)
- atom\_setparse
  - atom, [311](#)
- atom\_setsym
  - atom, [312](#)
- atom\_setsym\_array
  - atom, [312](#)
- atomarray
  - atomarray\_appendatom, [214](#)
  - atomarray\_appendatoms, [214](#)
  - atomarray\_chuckindex, [214](#)
  - atomarray\_clear, [215](#)
  - atomarray\_copyatoms, [215](#)
  - atomarray\_duplicate, [215](#)
  - atomarray\_funall, [216](#)
  - atomarray\_getatoms, [216](#)
  - atomarray\_getindex, [217](#)
  - atomarray\_getsize, [217](#)
  - atomarray\_new, [217](#)
  - atomarray\_setatoms, [218](#)
- atomarray\_appendatom
  - atomarray, [214](#)
- atomarray\_appendatoms
  - atomarray, [214](#)
- atomarray\_chuckindex
  - atomarray, [214](#)
- atomarray\_clear
  - atomarray, [215](#)
- atomarray\_copyatoms
  - atomarray, [215](#)
- atomarray\_duplicate
  - atomarray, [215](#)
- atomarray\_funall
  - atomarray, [216](#)
- atomarray\_getatoms
  - atomarray, [216](#)
- atomarray\_getindex
  - atomarray, [217](#)
- atomarray\_getsize
  - atomarray, [217](#)
- atomarray\_new
  - atomarray, [217](#)
- atomarray\_setatoms
  - atomarray, [218](#)
- atombuf
  - atombuf\_free, [314](#)
  - atombuf\_new, [314](#)
  - atombuf\_text, [315](#)
- atombuf\_free
- atombuf, [314](#)
- atombuf\_new
  - atombuf, [314](#)
- atombuf\_text
  - atombuf, [315](#)
- Atombufs, [314](#)
- ATOMIC\_DECREMENT
  - threading, [516](#)
- ATOMIC\_INCREMENT
  - threading, [516](#)
- atomisatomarray
  - atom, [312](#)
- atomisdictionary
  - atom, [313](#)
- atomisstring
  - atom, [313](#)
- Atoms, [295](#)
- atoms\_to\_jrgba
  - color, [567](#)
- attr
  - ATTR\_FLAGS\_NONE, [169](#)
  - ATTR\_GET\_DEFER, [169](#)
  - ATTR\_GET\_DEFER\_LOW, [169](#)
  - ATTR\_GET\_OPAQUE, [169](#)
  - ATTR\_GET\_OPAQUE\_USER, [169](#)
  - ATTR\_GET\_USURP, [169](#)
  - ATTR\_GET\_USURP\_LOW, [169](#)
  - ATTR\_SET\_DEFER, [169](#)
  - ATTR\_SET\_DEFER\_LOW, [169](#)
  - ATTR\_SET\_OPAQUE, [169](#)
  - ATTR\_SET\_OPAQUE\_USER, [169](#)
  - ATTR\_SET\_USURP, [169](#)
  - ATTR\_SET\_USURP\_LOW, [169](#)
  - attr\_addfilter\_clip, [169](#)
  - attr\_addfilter\_clip\_scale, [170](#)
  - attr\_addfilterget\_clip, [170](#)
  - attr\_addfilterget\_clip\_scale, [171](#)
  - attr\_addfilterget\_proc, [171](#)
  - attr\_addfilterset\_clip, [171](#)
  - attr\_addfilterset\_clip\_scale, [172](#)
  - attr\_addfilterset\_proc, [172](#)
  - attr\_args\_dictionary, [173](#)
  - attr\_args\_offset, [173](#)
  - attr\_args\_process, [174](#)
  - attr\_dictionary\_process, [174](#)
  - attr\_offset\_array\_new, [174](#)
  - attr\_offset\_new, [175](#)
  - attribute\_new, [176](#)
  - CLASS\_ATTR\_ACCESSORS, [132](#)
  - CLASS\_ATTR\_ADD\_FLAGS, [132](#)
  - CLASS\_ATTR\_ALIAS, [132](#)
  - CLASS\_ATTR\_ATOM, [133](#)
  - CLASS\_ATTR\_ATOM\_ARRAY, [133](#)
  - CLASS\_ATTR\_ATOM\_VARSIZE, [133](#)

- CLASS\_ATTR\_CATEGORY, 134
- CLASS\_ATTR\_CHAR, 134
- CLASS\_ATTR\_CHAR\_ARRAY, 134
- CLASS\_ATTR\_CHAR\_VARSIZE, 135
- CLASS\_ATTR\_DEFAULT, 135
- CLASS\_ATTR\_DEFAULT\_PAINT, 136
- CLASS\_ATTR\_DEFAULT\_SAVE, 136
- CLASS\_ATTR\_DEFAULT\_SAVE\_PAINT, 136
- CLASS\_ATTR\_DEFAULTNAME, 137
- CLASS\_ATTR\_DEFAULTNAME\_PAINT, 137
- CLASS\_ATTR\_DEFAULTNAME\_SAVE, 137
- CLASS\_ATTR\_DEFAULTNAME\_SAVE\_PAINT, 138
- CLASS\_ATTR\_DOUBLE, 138
- CLASS\_ATTR\_DOUBLE\_ARRAY, 139
- CLASS\_ATTR\_DOUBLE\_VARSIZE, 139
- CLASS\_ATTR\_ENUM, 139
- CLASS\_ATTR\_ENUMINDEX, 140
- CLASS\_ATTR\_FILTER\_CLIP, 140
- CLASS\_ATTR\_FILTER\_MAX, 141
- CLASS\_ATTR\_FILTER\_MIN, 141
- CLASS\_ATTR\_FLOAT, 142
- CLASS\_ATTR\_FLOAT\_ARRAY, 142
- CLASS\_ATTR\_FLOAT\_VARSIZE, 142
- CLASS\_ATTR\_INVISIBLE, 143
- CLASS\_ATTR\_LABEL, 143
- CLASS\_ATTR\_LONG, 143
- CLASS\_ATTR\_LONG\_ARRAY, 144
- CLASS\_ATTR\_LONG\_VARSIZE, 144
- CLASS\_ATTR\_MAX, 145
- CLASS\_ATTR\_MIN, 145
- CLASS\_ATTR\_OBJ, 145
- CLASS\_ATTR\_OBJ\_ARRAY, 146
- CLASS\_ATTR\_OBJ\_VARSIZE, 146
- CLASS\_ATTR\_ORDER, 147
- CLASS\_ATTR\_PAINT, 147
- CLASS\_ATTR\_REMOVE\_FLAGS, 147
- CLASS\_ATTR\_RGBA, 148
- CLASS\_ATTR\_SAVE, 148
- CLASS\_ATTR\_STYLE, 148
- CLASS\_ATTR\_STYLE\_LABEL, 149
- CLASS\_ATTR\_SYM, 149
- CLASS\_ATTR\_SYM\_ARRAY, 150
- CLASS\_ATTR\_SYM\_VARSIZE, 150
- CLASS\_STICKY\_ATTR, 150
- CLASS\_STICKY\_ATTR\_CLEAR, 151
- CLASS\_STICKY\_METHOD, 151
- CLASS\_STICKY\_METHOD\_CLEAR, 152
- e\_max\_attrflags, 169
- OBJ\_ATTR\_ATOM, 152
- OBJ\_ATTR\_ATOM\_ARRAY, 153
- OBJ\_ATTR\_CHAR, 153
- OBJ\_ATTR\_CHAR\_ARRAY, 153
- OBJ\_ATTR\_DEFAULT, 153
- OBJ\_ATTR\_DEFAULT\_SAVE, 154
- OBJ\_ATTR\_DOUBLE, 154
- OBJ\_ATTR\_DOUBLE\_ARRAY, 154
- OBJ\_ATTR\_FLOAT, 155
- OBJ\_ATTR\_FLOAT\_ARRAY, 155
- OBJ\_ATTR\_LONG, 155
- OBJ\_ATTR\_LONG\_ARRAY, 156
- OBJ\_ATTR\_OBJ, 156
- OBJ\_ATTR\_OBJ\_ARRAY, 156
- OBJ\_ATTR\_SAVE, 156
- OBJ\_ATTR\_SYM, 157
- OBJ\_ATTR\_SYM\_ARRAY, 157
- object\_addattr, 177
- object\_attr\_get, 177
- object\_attr\_get\_rect, 178
- object\_attr\_getchar\_array, 178
- object\_attr\_getcolor, 178
- object\_attr\_getdouble\_array, 179
- object\_attr\_getdump, 179
- object\_attr\_getfloat, 179
- object\_attr\_getfloat\_array, 180
- object\_attr\_getjrgba, 180
- object\_attr\_getlong, 180
- object\_attr\_getlong\_array, 181
- object\_attr\_getpt, 181
- object\_attr\_getsize, 181
- object\_attr\_getsym, 182
- object\_attr\_getsym\_array, 182
- object\_attr\_method, 182
- object\_attr\_set\_rect, 183
- object\_attr\_setchar\_array, 183
- object\_attr\_setcolor, 183
- object\_attr\_setdouble\_array, 184
- object\_attr\_setfloat, 184
- object\_attr\_setfloat\_array, 184
- object\_attr\_setjrgba, 185
- object\_attr\_setlong, 185
- object\_attr\_setlong\_array, 185
- object\_attr\_setparse, 186
- object\_attr\_setpt, 186
- object\_attr\_setsize, 186
- object\_attr\_setsym, 187
- object\_attr\_setsym\_array, 187
- object\_attr\_setvalueof, 187
- object\_attr\_usercanget, 188
- object\_attr\_usercanset, 188
- object\_chuckattr, 188
- object\_deleteattr, 189
- object\_new\_parse, 189
- STATIC\_ATTR\_ATOM, 157
- STATIC\_ATTR\_ATOM\_ARRAY, 158

- STATIC\_ATTR\_CHAR, 158
- STATIC\_ATTR\_CHAR\_ARRAY, 158
- STATIC\_ATTR\_DOUBLE, 158
- STATIC\_ATTR\_DOUBLE\_ARRAY, 159
- STATIC\_ATTR\_FLOAT, 159
- STATIC\_ATTR\_FLOAT\_ARRAY, 159
- STATIC\_ATTR\_LONG, 160
- STATIC\_ATTR\_LONG\_ARRAY, 160
- STATIC\_ATTR\_OBJ, 160
- STATIC\_ATTR\_OBJ\_ARRAY, 160
- STATIC\_ATTR\_SYM, 161
- STATIC\_ATTR\_SYM\_ARRAY, 161
- STRUCT\_ATTR\_ATOM, 161
- STRUCT\_ATTR\_ATOM\_ARRAY, 162
- STRUCT\_ATTR\_ATOM\_VARSIZE, 162
- STRUCT\_ATTR\_CHAR, 162
- STRUCT\_ATTR\_CHAR\_ARRAY, 163
- STRUCT\_ATTR\_CHAR\_VARSIZE, 163
- STRUCT\_ATTR\_DOUBLE, 163
- STRUCT\_ATTR\_DOUBLE\_ARRAY, 164
- STRUCT\_ATTR\_DOUBLE\_VARSIZE, 164
- STRUCT\_ATTR\_FLOAT, 164
- STRUCT\_ATTR\_FLOAT\_ARRAY, 165
- STRUCT\_ATTR\_FLOAT\_VARSIZE, 165
- STRUCT\_ATTR\_LONG, 165
- STRUCT\_ATTR\_LONG\_ARRAY, 166
- STRUCT\_ATTR\_LONG\_VARSIZE, 166
- STRUCT\_ATTR\_OBJ, 166
- STRUCT\_ATTR\_OBJ\_ARRAY, 167
- STRUCT\_ATTR\_OBJ\_VARSIZE, 167
- STRUCT\_ATTR\_SYM, 167
- STRUCT\_ATTR\_SYM\_ARRAY, 168
- STRUCT\_ATTR\_SYM\_VARSIZE, 168
- ATTR\_FLAGS\_NONE
  - attr, 169
- ATTR\_GET\_DEFER
  - attr, 169
- ATTR\_GET\_DEFER\_LOW
  - attr, 169
- ATTR\_GET\_OPAQUE
  - attr, 169
- ATTR\_GET\_OPAQUE\_USER
  - attr, 169
- ATTR\_GET\_USURP
  - attr, 169
- ATTR\_GET\_USURP\_LOW
  - attr, 169
- ATTR\_SET\_DEFER
  - attr, 169
- ATTR\_SET\_DEFER\_LOW
  - attr, 169
- ATTR\_SET\_OPAQUE
  - attr, 169
- ATTR\_SET\_OPAQUE\_USER
  - attr, 169
- ATTR\_SET\_USURP
  - attr, 169
- ATTR\_SET\_USURP\_LOW
  - attr, 169
- attr\_addfilter\_clip
  - attr, 169
- attr\_addfilter\_clip\_scale
  - attr, 170
- attr\_addfilterget\_clip
  - attr, 170
- attr\_addfilterget\_clip\_scale
  - attr, 171
- attr\_addfilterget\_proc
  - attr, 171
- attr\_addfilterset\_clip
  - attr, 171
- attr\_addfilterset\_clip\_scale
  - attr, 172
- attr\_addfilterset\_proc
  - attr, 172
- attr\_args\_dictionary
  - attr, 173
- attr\_args\_offset
  - attr, 173
- attr\_args\_process
  - attr, 174
- attr\_dictionary\_process
  - attr, 174
- attr\_offset\_array\_new
  - attr, 174
- attr\_offset\_new
  - attr, 175
- attribute\_new
  - attr, 176
- Attributes, 117
- bangout
  - inout, 204
- BEGIN\_USING\_C\_LINKAGE
  - misc, 358
- binbuf
  - binbuf\_append, 317
  - binbuf\_eval, 317
  - binbuf\_getatom, 317
  - binbuf\_insert, 317
  - binbuf\_new, 318
  - binbuf\_set, 318
  - binbuf\_text, 318
  - binbuf\_totext, 319
  - binbuf\_vinsert, 319
  - readatom, 320
- binbuf\_append
  - binbuf, 317

- binbuf\_eval
  - binbuf, [317](#)
- binbuf\_getatom
  - binbuf, [317](#)
- binbuf\_insert
  - binbuf, [317](#)
- binbuf\_new
  - binbuf, [318](#)
- binbuf\_set
  - binbuf, [318](#)
- binbuf\_text
  - binbuf, [318](#)
- binbuf\_totext
  - binbuf, [319](#)
- binbuf\_vinsert
  - binbuf, [319](#)
- Binbufs, [316](#)
- Box Layer, [588](#)
- boxlayer
  - jbox\_end\_layer, [589](#)
  - jbox\_invalidate\_layer, [589](#)
  - jbox\_paint\_layer, [589](#)
  - jbox\_start\_layer, [589](#)
- Buffers, [409](#)
- Byte Ordering, [371](#)
- byteorder
  - BYTEORDER\_LSBF32, [372](#)
  - BYTEORDER\_LSBF64, [372](#)
  - BYTEORDER\_LSBW16, [372](#)
  - BYTEORDER\_LSBW32, [373](#)
  - BYTEORDER\_MSBF32, [373](#)
  - BYTEORDER\_MSBF64, [373](#)
  - BYTEORDER\_MSBW16, [374](#)
  - BYTEORDER\_MSBW32, [374](#)
  - BYTEORDER\_SWAPF32, [374](#)
  - BYTEORDER\_SWAPF64, [374](#)
  - BYTEORDER\_SWAPW16, [375](#)
  - BYTEORDER\_SWAPW32, [375](#)
  - C74\_BIG\_ENDIAN, [375](#)
  - C74\_LITTLE\_ENDIAN, [375](#)
- BYTEORDER\_LSBF32
  - byteorder, [372](#)
- BYTEORDER\_LSBF64
  - byteorder, [372](#)
- BYTEORDER\_LSBW16
  - byteorder, [372](#)
- BYTEORDER\_LSBW32
  - byteorder, [373](#)
- BYTEORDER\_MSBF32
  - byteorder, [373](#)
- BYTEORDER\_MSBF64
  - byteorder, [373](#)
- BYTEORDER\_MSBW16
  - byteorder, [374](#)
- BYTEORDER\_MSBW32
  - byteorder, [374](#)
- BYTEORDER\_SWAPF32
  - byteorder, [374](#)
- BYTEORDER\_SWAPF64
  - byteorder, [374](#)
- BYTEORDER\_SWAPW16
  - byteorder, [375](#)
- BYTEORDER\_SWAPW32
  - byteorder, [375](#)
- C74\_BIG\_ENDIAN
  - byteorder, [375](#)
- C74\_LITTLE\_ENDIAN
  - byteorder, [375](#)
- calcoffset
  - misc, [358](#)
- charset\_convert
  - unicode, [594](#)
- charset\_unicondetoutf8
  - unicode, [594](#)
- charset\_utf8\_count
  - unicode, [595](#)
- charset\_utf8\_offset
  - unicode, [595](#)
- charset\_utf8tounicode
  - unicode, [595](#)
- class
  - CLASS\_FLAG\_ALIAS, [192](#)
  - CLASS\_FLAG\_BOX, [192](#)
  - CLASS\_FLAG\_DO\_NOT\_PARSE\_ATTR\_ARGS, [192](#)
  - CLASS\_FLAG\_NEWDICTIONARY, [192](#)
  - CLASS\_FLAG\_NOATTRIBUTES, [192](#)
  - CLASS\_FLAG\_OWNATTRIBUTES, [192](#)
  - CLASS\_FLAG\_POLYGLOT, [192](#)
  - CLASS\_FLAG\_REGISTERED, [192](#)
  - CLASS\_FLAG\_SCHED\_PURGE, [192](#)
  - CLASS\_FLAG\_UIOBJECT, [192](#)
  - class\_addattr, [193](#)
  - class\_addmethod, [193](#)
  - class\_alias, [193](#)
  - CLASS\_BOX, [192](#)
  - class\_dumpout\_wrap, [193](#)
  - class\_findbyname, [194](#)
  - class\_findbyname\_casefree, [194](#)
  - class\_free, [194](#)
  - class\_is\_ui, [195](#)
  - class\_nameget, [195](#)
  - class\_new, [195](#)
  - class\_obexoffset\_get, [195](#)
  - class\_obexoffset\_set, [196](#)
  - class\_register, [196](#)
  - e\_max\_class\_flags, [192](#)

- CLASS\_FLAG\_ALIAS
  - class, [192](#)
- CLASS\_FLAG\_BOX
  - class, [192](#)
- CLASS\_FLAG\_DO\_NOT\_PARSE\_ATTR\_ARGS
  - class, [192](#)
- CLASS\_FLAG\_NEWDICTIONARY
  - class, [192](#)
- CLASS\_FLAG\_NOATTRIBUTES
  - class, [192](#)
- CLASS\_FLAG\_OWNATTRIBUTES
  - class, [192](#)
- CLASS\_FLAG\_POLYGLOT
  - class, [192](#)
- CLASS\_FLAG\_REGISTERED
  - class, [192](#)
- CLASS\_FLAG\_SCHED\_PURGE
  - class, [192](#)
- CLASS\_FLAG\_UIOBJECT
  - class, [192](#)
- class\_addattr
  - class, [193](#)
- class\_addmethod
  - class, [193](#)
- class\_alias
  - class, [193](#)
- CLASS\_ATTR\_ACCESSORS
  - attr, [132](#)
- CLASS\_ATTR\_ADD\_FLAGS
  - attr, [132](#)
- CLASS\_ATTR\_ALIAS
  - attr, [132](#)
- CLASS\_ATTR\_ATOM
  - attr, [133](#)
- CLASS\_ATTR\_ATOM\_ARRAY
  - attr, [133](#)
- CLASS\_ATTR\_ATOM\_VARSIZE
  - attr, [133](#)
- CLASS\_ATTR\_CATEGORY
  - attr, [134](#)
- CLASS\_ATTR\_CHAR
  - attr, [134](#)
- CLASS\_ATTR\_CHAR\_ARRAY
  - attr, [134](#)
- CLASS\_ATTR\_CHAR\_VARSIZE
  - attr, [135](#)
- CLASS\_ATTR\_DEFAULT
  - attr, [135](#)
- CLASS\_ATTR\_DEFAULT\_PAINT
  - attr, [136](#)
- CLASS\_ATTR\_DEFAULT\_SAVE
  - attr, [136](#)
- CLASS\_ATTR\_DEFAULT\_SAVE\_PAINT
  - attr, [136](#)
- CLASS\_ATTR\_DEFAULTNAME
  - attr, [137](#)
- CLASS\_ATTR\_DEFAULTNAME\_PAINT
  - attr, [137](#)
- CLASS\_ATTR\_DEFAULTNAME\_SAVE
  - attr, [137](#)
- CLASS\_ATTR\_DEFAULTNAME\_SAVE\_PAINT
  - attr, [138](#)
- CLASS\_ATTR\_DOUBLE
  - attr, [138](#)
- CLASS\_ATTR\_DOUBLE\_ARRAY
  - attr, [139](#)
- CLASS\_ATTR\_DOUBLE\_VARSIZE
  - attr, [139](#)
- CLASS\_ATTR\_ENUM
  - attr, [139](#)
- CLASS\_ATTR\_ENUMINDEX
  - attr, [140](#)
- CLASS\_ATTR\_FILTER\_CLIP
  - attr, [140](#)
- CLASS\_ATTR\_FILTER\_MAX
  - attr, [141](#)
- CLASS\_ATTR\_FILTER\_MIN
  - attr, [141](#)
- CLASS\_ATTR\_FLOAT
  - attr, [142](#)
- CLASS\_ATTR\_FLOAT\_ARRAY
  - attr, [142](#)
- CLASS\_ATTR\_FLOAT\_VARSIZE
  - attr, [142](#)
- CLASS\_ATTR\_INVISIBLE
  - attr, [143](#)
- CLASS\_ATTR\_LABEL
  - attr, [143](#)
- CLASS\_ATTR\_LONG
  - attr, [143](#)
- CLASS\_ATTR\_LONG\_ARRAY
  - attr, [144](#)
- CLASS\_ATTR\_LONG\_VARSIZE
  - attr, [144](#)
- CLASS\_ATTR\_MAX
  - attr, [145](#)
- CLASS\_ATTR\_MIN
  - attr, [145](#)
- CLASS\_ATTR\_OBJ
  - attr, [145](#)
- CLASS\_ATTR\_OBJ\_ARRAY
  - attr, [146](#)
- CLASS\_ATTR\_OBJ\_VARSIZE
  - attr, [146](#)
- CLASS\_ATTR\_ORDER
  - attr, [147](#)
- CLASS\_ATTR\_PAINT
  - attr, [147](#)

- CLASS\_ATTR\_REMOVE\_FLAGS
  - attr, [147](#)
- CLASS\_ATTR\_RGBA
  - attr, [148](#)
- CLASS\_ATTR\_SAVE
  - attr, [148](#)
- CLASS\_ATTR\_STYLE
  - attr, [148](#)
- CLASS\_ATTR\_STYLE\_LABEL
  - attr, [149](#)
- CLASS\_ATTR\_SYM
  - attr, [149](#)
- CLASS\_ATTR\_SYM\_ARRAY
  - attr, [150](#)
- CLASS\_ATTR\_SYM\_VARSIZE
  - attr, [150](#)
- CLASS\_BOX
  - class, [192](#)
- class\_dspinit
  - msp, [405](#)
- class\_dspinitbox
  - msp, [405](#)
- class\_dspinitjbox
  - msp, [406](#)
- class\_dumpout\_wrap
  - class, [193](#)
- class\_findbyname
  - class, [194](#)
- class\_findbyname\_casefree
  - class, [194](#)
- class\_free
  - class, [194](#)
- class\_is\_ui
  - class, [195](#)
- class\_nameget
  - class, [195](#)
- class\_new
  - class, [195](#)
- class\_obexoffset\_get
  - class, [195](#)
- class\_obexoffset\_set
  - class, [196](#)
- class\_old
  - addbang, [198](#)
  - addfloat, [198](#)
  - addftx, [198](#)
  - addint, [198](#)
  - addinx, [198](#)
  - addmess, [199](#)
  - alias, [199](#)
  - class\_setname, [199](#)
  - egetfn, [199](#)
  - freeobject, [200](#)
  - getfn, [200](#)
  - newinstance, [200](#)
  - newobject, [201](#)
  - setup, [201](#)
  - typedmess, [202](#)
  - zgetfn, [202](#)
- class\_register
  - class, [196](#)
- class\_setname
  - class\_old, [199](#)
- CLASS\_STICKY\_ATTR
  - attr, [150](#)
- CLASS\_STICKY\_ATTR\_CLEAR
  - attr, [151](#)
- CLASS\_STICKY\_METHOD
  - attr, [151](#)
- CLASS\_STICKY\_METHOD\_CLEAR
  - attr, [152](#)
- class\_time\_addattr
  - time, [504](#)
- Classes, [190](#)
- classname\_openhelp
  - obj, [417](#)
- classname\_openquery
  - obj, [417](#)
- classname\_openrefpage
  - obj, [417](#)
- CLIP
  - misc, [358](#)
- clock\_delay
  - clocks, [488](#)
- clock\_fdelay
  - clocks, [488](#)
- clock\_getftime
  - clocks, [488](#)
- clock\_new
  - clocks, [489](#)
- clock\_unset
  - clocks, [489](#)
- Clocks, [484](#)
- clocks
  - clock\_delay, [488](#)
  - clock\_fdelay, [488](#)
  - clock\_getftime, [488](#)
  - clock\_new, [489](#)
  - clock\_unset, [489](#)
  - gettime, [489](#)
  - scheduler\_gettime, [489](#)
  - scheduler\_new, [490](#)
  - scheduler\_run, [490](#)
  - scheduler\_set, [490](#)
  - scheduler\_settime, [490](#)
  - setclock\_delay, [491](#)
  - setclock\_fdelay, [491](#)
  - setclock\_getftime, [491](#)

- setclock\_gettime, [492](#)
- setclock\_unset, [492](#)
- sys timer\_gettime, [492](#)
- color
  - atoms\_to\_jrgba, [567](#)
  - jrgba\_attr\_get, [568](#)
  - jrgba\_attr\_set, [568](#)
  - jrgba\_compare, [568](#)
  - jrgba\_copy, [569](#)
  - jrgba\_set, [569](#)
  - jrgba\_to\_atoms, [569](#)
- Colors, [567](#)
- Console, [366](#)
- console
  - cpost, [366](#)
  - error, [367](#)
  - object\_error, [367](#)
  - object\_error\_obtrusive, [368](#)
  - object\_post, [368](#)
  - object\_warn, [368](#)
  - ouchstring, [369](#)
  - post, [369](#)
  - postatom, [370](#)
- cpost
  - console, [366](#)
- critical
  - critical\_enter, [523](#)
  - critical\_exit, [524](#)
  - critical\_free, [524](#)
  - critical\_new, [524](#)
  - critical\_tryenter, [524](#)
- Critical Regions, [522](#)
- critical\_enter
  - critical, [523](#)
- critical\_exit
  - critical, [524](#)
- critical\_free
  - critical, [524](#)
- critical\_new
  - critical, [524](#)
- critical\_tryenter
  - critical, [524](#)
- Data Storage, [210](#)
- Data Types, [293](#)
- Database, [219](#)
- database
  - db\_close, [221](#)
  - db\_open, [222](#)
  - db\_query, [222](#)
  - db\_query\_getlastinsertid, [223](#)
  - db\_query\_silent, [223](#)
  - db\_query\_table\_addcolumn, [224](#)
  - db\_query\_table\_new, [225](#)
  - db\_result\_clear, [225](#)
  - db\_result\_datetimeinseconds, [225](#)
  - db\_result\_fieldname, [226](#)
  - db\_result\_float, [226](#)
  - db\_result\_long, [227](#)
  - db\_result\_nextrecord, [227](#)
  - db\_result\_numfields, [228](#)
  - db\_result\_numrecords, [228](#)
  - db\_result\_reset, [228](#)
  - db\_result\_string, [229](#)
  - db\_transaction\_end, [229](#)
  - db\_transaction\_flush, [230](#)
  - db\_transaction\_start, [230](#)
  - db\_util\_datetostring, [230](#)
  - db\_util\_stringtodate, [231](#)
  - db\_view\_create, [231](#)
  - db\_view\_getresult, [232](#)
  - db\_view\_remove, [232](#)
  - db\_view\_setquery, [233](#)
  - t\_database, [221](#)
  - t\_db\_result, [221](#)
  - t\_db\_view, [221](#)
- datastore
  - e\_max\_datastore\_flags, [212](#)
  - OBJ\_FLAG\_DATA, [212](#)
  - OBJ\_FLAG\_MEMORY, [212](#)
  - OBJ\_FLAG\_OBJ, [212](#)
  - OBJ\_FLAG\_REF, [212](#)
  - OBJ\_FLAG\_SILENT, [212](#)
  - t\_cmpfn, [211](#)
- datatypes
  - t\_max\_err, [294](#)
- DataGridView, [591](#)
- db\_close
  - database, [221](#)
- db\_open
  - database, [222](#)
- db\_query
  - database, [222](#)
- db\_query\_getlastinsertid
  - database, [223](#)
- db\_query\_silent
  - database, [223](#)
- db\_query\_table\_addcolumn
  - database, [224](#)
- db\_query\_table\_new
  - database, [225](#)
- db\_result\_clear
  - database, [225](#)
- db\_result\_datetimeinseconds
  - database, [225](#)
- db\_result\_fieldname
  - database, [226](#)
- db\_result\_float



- database, 226
- db\_result\_long
  - database, 227
- db\_result\_nextrecord
  - database, 227
- db\_result\_numfields
  - database, 228
- db\_result\_numrecords
  - database, 228
- db\_result\_reset
  - database, 228
- db\_result\_string
  - database, 229
- db\_transaction\_end
  - database, 229
- db\_transaction\_flush
  - database, 230
- db\_transaction\_start
  - database, 230
- db\_util\_datetostring
  - database, 230
- db\_util\_stringtodate
  - database, 231
- db\_view\_create
  - database, 231
- db\_view\_getresult
  - database, 232
- db\_view\_remove
  - database, 232
- db\_view\_setquery
  - database, 233
- defer
  - threading, 517
- defer\_low
  - threading, 517
- Dictionary, 234
- dictionary
  - dictionary\_appendatom, 239
  - dictionary\_appendatomarray, 239
  - dictionary\_appendatoms, 240
  - dictionary\_appenddictionary, 240
  - dictionary\_appendfloat, 240
  - dictionary\_appendlong, 241
  - dictionary\_appendobject, 241
  - dictionary\_appendstring, 241
  - dictionary\_appendsym, 242
  - dictionary\_chuckentry, 242
  - dictionary\_clear, 242
  - dictionary\_copyatoms, 243
  - dictionary\_copydefatoms, 243
  - dictionary\_copyentries, 244
  - dictionary\_copyunique, 244
  - dictionary\_deleteentry, 244
  - dictionary\_dump, 245
  - dictionary\_entry\_getkey, 245
  - dictionary\_entry\_getvalue, 245
  - dictionary\_entryisatomarray, 246
  - dictionary\_entryisdictionary, 246
  - dictionary\_entryisstring, 246
  - dictionary\_freekeys, 246
  - dictionary\_funall, 247
  - dictionary\_getatom, 247
  - dictionary\_getatomarray, 248
  - dictionary\_getatoms, 248
  - dictionary\_getdefatom, 248
  - dictionary\_getdefatoms, 249
  - dictionary\_getdeffloat, 249
  - dictionary\_getdeflong, 249
  - dictionary\_getdefstring, 250
  - dictionary\_getdefsym, 250
  - dictionary\_getdictionary, 251
  - dictionary\_getentrycount, 251
  - dictionary\_getfloat, 251
  - dictionary\_getkeys, 251
  - dictionary\_getlong, 252
  - dictionary\_getobject, 252
  - dictionary\_getstring, 253
  - dictionary\_getsym, 253
  - dictionary\_hasentry, 253
  - dictionary\_new, 253
  - dictionary\_read, 254
  - dictionary\_sprintf, 254
  - dictionary\_write, 255
  - postdictionary, 255
- dictionary\_appendatom
  - dictionary, 239
- dictionary\_appendatomarray
  - dictionary, 239
- dictionary\_appendatoms
  - dictionary, 240
- dictionary\_appenddictionary
  - dictionary, 240
- dictionary\_appendfloat
  - dictionary, 240
- dictionary\_appendlong
  - dictionary, 241
- dictionary\_appendobject
  - dictionary, 241
- dictionary\_appendstring
  - dictionary, 241
- dictionary\_appendsym
  - dictionary, 242
- dictionary\_chuckentry
  - dictionary, 242
- dictionary\_clear
  - dictionary, 242
- dictionary\_copyatoms
  - dictionary, 243

- dictionary\_copydefatoms
  - dictionary, [243](#)
- dictionary\_copyentries
  - dictionary, [244](#)
- dictionary\_copyunique
  - dictionary, [244](#)
- dictionary\_deleteentry
  - dictionary, [244](#)
- dictionary\_dump
  - dictionary, [245](#)
- dictionary\_entry\_getkey
  - dictionary, [245](#)
- dictionary\_entry\_getvalue
  - dictionary, [245](#)
- dictionary\_entryisatomarray
  - dictionary, [246](#)
- dictionary\_entryisdictionary
  - dictionary, [246](#)
- dictionary\_entryisstring
  - dictionary, [246](#)
- dictionary\_freekeys
  - dictionary, [246](#)
- dictionary\_funall
  - dictionary, [247](#)
- dictionary\_getatom
  - dictionary, [247](#)
- dictionary\_getatomarray
  - dictionary, [248](#)
- dictionary\_getatoms
  - dictionary, [248](#)
- dictionary\_getdefatom
  - dictionary, [248](#)
- dictionary\_getdefatom
  - dictionary, [249](#)
- dictionary\_getdeffloat
  - dictionary, [249](#)
- dictionary\_getdeflong
  - dictionary, [249](#)
- dictionary\_getdefstring
  - dictionary, [250](#)
- dictionary\_getdefsym
  - dictionary, [250](#)
- dictionary\_getdictionary
  - dictionary, [251](#)
- dictionary\_getentrycount
  - dictionary, [251](#)
- dictionary\_getfloat
  - dictionary, [251](#)
- dictionary\_getkeys
  - dictionary, [251](#)
- dictionary\_getlong
  - dictionary, [252](#)
- dictionary\_getobject
  - dictionary, [252](#)
- dictionary\_getstring
  - dictionary, [253](#)
- dictionary\_getsym
  - dictionary, [253](#)
- dictionary\_hasentry
  - dictionary, [253](#)
- dictionary\_new
  - dictionary, [253](#)
- dictionary\_read
  - dictionary, [254](#)
- dictionary\_sprintf
  - dictionary, [254](#)
- dictionary\_write
  - dictionary, [255](#)
- disposhandle
  - memory, [349](#)
- dsp\_add
  - misp, [406](#)
- dsp\_addv
  - misp, [406](#)
- e\_max\_atom\_gettext\_flags
  - atom, [298](#)
- e\_max\_atomtypes
  - atom, [299](#)
- e\_max\_attrflags
  - attr, [169](#)
- e\_max\_class\_flags
  - class, [192](#)
- e\_max\_datastore\_flags
  - datastore, [212](#)
- e\_max\_dateflags
  - sysstime, [497](#)
- e\_max\_errorcodes
  - misc, [360](#)
- e\_max\_expr\_types
  - expr, [378](#)
- e\_max\_fileinfo\_flags
  - files, [330](#)
- e\_max\_openfile\_permissions
  - files, [330](#)
- e\_max\_path\_folder\_flags
  - files, [331](#)
- e\_max\_path\_styles
  - files, [331](#)
- e\_max\_path\_types
  - files, [331](#)
- e\_max\_sysfile\_posmodes
  - files, [332](#)
- e\_max\_sysfile\_textflags
  - files, [332](#)
- e\_max\_systhread\_mutex\_flags
  - threading, [516](#)
- e\_max\_wind\_advise\_result

- misc, [360](#)
- eAltKey
  - jmouse, [397](#)
- eAutoRepeat
  - jmouse, [397](#)
- eCapsLock
  - jmouse, [397](#)
- eCommandKey
  - jmouse, [397](#)
- eControlKey
  - jmouse, [397](#)
- egetfn
  - class\_old, [199](#)
- eLeftButton
  - jmouse, [397](#)
- eMiddleButton
  - jmouse, [397](#)
- ePopupMenu
  - jmouse, [397](#)
- eRightButton
  - jmouse, [397](#)
- error
  - console, [367](#)
- error\_subscribe
  - misc, [360](#)
- error\_sym
  - misc, [360](#)
- error\_unsubscribe
  - misc, [361](#)
- eShiftKey
  - jmouse, [397](#)
- ET\_FI
  - expr, [378](#)
- ET\_FLT
  - expr, [378](#)
- ET\_FUNC
  - expr, [378](#)
- ET\_II
  - expr, [378](#)
- ET\_INT
  - expr, [378](#)
- ET\_LB
  - expr, [378](#)
- ET\_LP
  - expr, [378](#)
- ET\_OP
  - expr, [378](#)
- ET\_SI
  - expr, [378](#)
- ET\_STR
  - expr, [378](#)
- ET\_SYM
  - expr, [378](#)
- ET\_TBL
  - expr, [378](#)
- expr, [378](#)
- ET\_VSYM
  - expr, [378](#)
- Event and File Serial Numbers, [387](#)
- evnum
  - evnum\_get, [387](#)
  - serialno, [387](#)
- evnum\_get
  - evnum, [387](#)
- Ex\_ex, [597](#)
- Example Projects, [399](#)
- expr
  - e\_max\_expr\_types, [378](#)
  - ET\_FI, [378](#)
  - ET\_FLT, [378](#)
  - ET\_FUNC, [378](#)
  - ET\_II, [378](#)
  - ET\_INT, [378](#)
  - ET\_LB, [378](#)
  - ET\_LP, [378](#)
  - ET\_OP, [378](#)
  - ET\_SI, [378](#)
  - ET\_STR, [378](#)
  - ET\_SYM, [378](#)
  - ET\_TBL, [378](#)
  - ET\_VSYM, [378](#)
  - expr\_eval, [379](#)
  - expr\_new, [379](#)
- expr\_eval
  - expr, [379](#)
- expr\_new
  - expr, [379](#)
- Extending expr, [377](#)
- fileload
  - loading\_max\_files, [389](#)
- files
  - e\_max\_fileinfo\_flags, [330](#)
  - e\_max\_openfile\_permissions, [330](#)
  - e\_max\_path\_folder\_flags, [331](#)
  - e\_max\_path\_styles, [331](#)
  - e\_max\_path\_types, [331](#)
  - e\_max\_sysfile\_posmodes, [332](#)
  - e\_max\_sysfile\_textflags, [332](#)
  - fileusage\_addfile, [332](#)
  - filewatcher\_new, [332](#)
  - locatefile, [333](#)
  - locatefile\_extended, [333](#)
  - locatefiletype, [334](#)
  - MAX\_FILENAME\_CHARS, [330](#)
  - open\_dialog, [335](#)
  - open\_promptset, [335](#)
  - PATH\_FILEINFO\_ALIAS, [330](#)
  - PATH\_FILEINFO\_FOLDER, [330](#)

- PATH\_FILEINFO\_PACKAGE, 330
- PATH\_FOLDER\_SNIFF, 331
- PATH\_READ\_PERM, 331
- PATH\_REPORTPACKAGEASFOLDER, 331
- PATH\_RW\_PERM, 331
- PATH\_STYLE\_COLON, 331
- PATH\_STYLE\_MAX, 331
- PATH\_STYLE\_NATIVE, 331
- PATH\_STYLE\_NATIVE\_WIN, 331
- PATH\_STYLE\_SLASH, 331
- PATH\_TYPE\_ABSOLUTE, 332
- PATH\_TYPE\_BOOT, 332
- PATH\_TYPE\_C74, 332
- PATH\_TYPE\_IGNORE, 332
- PATH\_TYPE\_PATH, 332
- PATH\_TYPE\_RELATIVE, 332
- PATH\_WRITE\_PERM, 331
- path\_closefolder, 335
- path\_createsysfile, 336
- path\_fileinfo, 336
- path\_foldernextfile, 336
- path\_frompathname, 337
- path\_getapppath, 337
- path\_getdefault, 337
- path\_getfilemoddate, 337
- path\_getmoddate, 337
- path\_nameconform, 338
- path\_openfolder, 338
- path\_opensysfile, 338
- path\_resolvefile, 339
- path\_setdefault, 339
- path\_topathname, 339
- path\_topotentialname, 340
- saveas\_dialog, 340
- saveas\_promptset, 340
- saveasdialog\_extended, 341
- SYSFILE\_ATMARK, 332
- SYSFILE\_FROMLEOF, 332
- SYSFILE\_FROMMARK, 332
- SYSFILE\_FROMSTART, 332
- sysfile\_close, 342
- sysfile\_geteof, 342
- sysfile\_getpos, 342
- sysfile\_openhandle, 343
- sysfile\_openptrsize, 343
- sysfile\_read, 343
- sysfile\_readtextfile, 344
- sysfile\_readtohandle, 344
- sysfile\_readtoptr, 344
- sysfile\_seteof, 345
- sysfile\_setpos, 345
- sysfile\_spoolcopy, 345
- sysfile\_write, 345
- sysfile\_writetextfile, 346
- t\_filehandle, 330
- TEXT\_ENCODING\_USE\_FILE, 332
- TEXT\_LB\_MAC, 332
- TEXT\_LB\_NATIVE, 332
- TEXT\_LB\_PC, 332
- TEXT\_LB\_UNIX, 332
- TEXT\_NULL\_TERMINATE, 332
- Files and Folders, 324
- fileusage\_addfile
  - files, 332
- filewatcher\_new
  - files, 332
- floatin
  - inout, 204
- floatout
  - inout, 204
- freebytes
  - memory, 349
- freebytes16
  - memory, 349
- freeobject
  - class\_old, 200
- gensym
  - symbol, 323
- getbytes
  - memory, 350
- getbytes16
  - memory, 350
- getfn
  - class\_old, 200
- gettime
  - clocks, 489
- globalsymbol\_bind
  - misc, 361
- globalsymbol\_dereference
  - misc, 361
- globalsymbol\_reference
  - misc, 361
- globalsymbol\_unbind
  - misc, 362
- growhandle
  - memory, 350
- Hash Table, 256
- HASH\_DEFSLOTS
  - hashtab, 258
- hashtab
  - HASH\_DEFSLOTS, 258
  - hashtab\_chuck, 258
  - hashtab\_chuckkey, 258
  - hashtab\_clear, 258
  - hashtab\_delete, 259
  - hashtab\_findfirst, 259

- hashtab\_flags, 260
- hashtab\_funall, 260
- hashtab\_getflags, 260
- hashtab\_getkeyflags, 260
- hashtab\_getkeys, 261
- hashtab\_getsize, 261
- hashtab\_keyflags, 262
- hashtab\_lookup, 262
- hashtab\_lookupflags, 262
- hashtab\_methodall, 263
- hashtab\_new, 263
- hashtab\_print, 263
- hashtab\_readonly, 264
- hashtab\_store, 264
- hashtab\_store\_safe, 264
- hashtab\_storeflags, 265
- hashtab\_chuck
  - hashtab, 258
- hashtab\_chuckkey
  - hashtab, 258
- hashtab\_clear
  - hashtab, 258
- hashtab\_delete
  - hashtab, 259
- hashtab\_findfirst
  - hashtab, 259
- hashtab\_flags
  - hashtab, 260
- hashtab\_funall
  - hashtab, 260
- hashtab\_getflags
  - hashtab, 260
- hashtab\_getkeyflags
  - hashtab, 260
- hashtab\_getkeys
  - hashtab, 261
- hashtab\_getsize
  - hashtab, 261
- hashtab\_keyflags
  - hashtab, 262
- hashtab\_lookup
  - hashtab, 262
- hashtab\_lookupflags
  - hashtab, 262
- hashtab\_methodall
  - hashtab, 263
- hashtab\_new
  - hashtab, 263
- hashtab\_print
  - hashtab, 263
- hashtab\_readonly
  - hashtab, 264
- hashtab\_store
  - hashtab, 264
- hashtab\_store\_safe
  - hashtab, 264
- hashtab\_storeflags
  - hashtab, 265
- HitBox
  - jbox, 458
- HitGrowBox
  - jbox, 458
- HitInlet
  - jbox, 458
- HitLine
  - jbox, 458
- HitNothing
  - jbox, 458
- HitOutlet
  - jbox, 458
- HitTestResult
  - jbox, 458
- Index Map, 266
- indexmap
  - indexmap\_append, 267
  - indexmap\_clear, 267
  - indexmap\_datafromindex, 267
  - indexmap\_delete, 267
  - indexmap\_delete\_index, 268
  - indexmap\_delete\_index\_multi, 268
  - indexmap\_delete\_multi, 268
  - indexmap\_getsize, 268
  - indexmap\_indexfromdata, 269
  - indexmap\_move, 269
  - indexmap\_new, 269
  - indexmap\_sort, 269
- indexmap\_append
  - indexmap, 267
- indexmap\_clear
  - indexmap, 267
- indexmap\_datafromindex
  - indexmap, 267
- indexmap\_delete
  - indexmap, 267
- indexmap\_delete\_index
  - indexmap, 268
- indexmap\_delete\_index\_multi
  - indexmap, 268
- indexmap\_delete\_multi
  - indexmap, 268
- indexmap\_getsize
  - indexmap, 268
- indexmap\_indexfromdata
  - indexmap, 269
- indexmap\_move
  - indexmap, 269
- indexmap\_new
  - indexmap, 269

- indexmap, 269
- indexmap\_sort
  - indexmap, 269
- inlet\_new
  - inout, 205
- Inlets and Outlets, 203
- inout
  - bangout, 204
  - floatin, 204
  - floatout, 204
  - inlet\_new, 205
  - intin, 205
  - intout, 205
  - listout, 206
  - outlet\_anything, 206
  - outlet\_bang, 207
  - outlet\_float, 207
  - outlet\_int, 207
  - outlet\_list, 207
  - outlet\_new, 208
  - proxy\_getinlet, 208
  - proxy\_new, 209
- InRange
  - misc, 359
- intin
  - inout, 205
- intload
  - loading\_max\_files, 389
- intout
  - inout, 205
- isr
  - threading, 518
- ITM Time Objects, 500
- itm\_barbeatunitstoticks
  - time, 504
- itm\_dereference
  - time, 504
- itm\_dump
  - time, 504
- itm\_getglobal
  - time, 505
- itm\_getname
  - time, 505
- itm\_getnamed
  - time, 505
- itm\_getresolution
  - time, 505
- itm\_getstate
  - time, 505
- itm\_getticks
  - time, 506
- itm\_gettime
  - time, 506
- itm\_gettimesignature
  - time, 506
- itm\_isunitfixed
  - time, 506
- itm\_mstosamps
  - time, 507
- itm\_mstoticks
  - time, 507
- itm\_pause
  - time, 507
- itm\_reference
  - time, 507
- itm\_resume
  - time, 508
- itm\_sampstoms
  - time, 508
- itm\_setresolution
  - time, 508
- itm\_settimesignature
  - time, 508
- itm\_tickstobarbeatunits
  - time, 508
- itm\_tickstoms
  - time, 509
- jbox, 452
  - HitBox, 458
  - HitGrowBox, 458
  - HitInlet, 458
  - HitLine, 458
  - HitNothing, 458
  - HitOutlet, 458
  - HitTestResult, 458
  - JBOX\_FONTFACE\_BOLD, 458
  - JBOX\_FONTFACE\_BOLDITALIC, 458
  - JBOX\_FONTFACE\_ITALIC, 458
  - JBOX\_FONTFACE\_REGULAR, 458
  - jbox\_free, 458
  - jbox\_get\_annotation, 458
  - jbox\_get\_background, 459
  - jbox\_get\_canhilite, 459
  - jbox\_get\_color, 459
  - jbox\_get\_drawfirstin, 459
  - jbox\_get\_drawinlast, 460
  - jbox\_get\_fontname, 460
  - jbox\_get\_fontsize, 460
  - jbox\_get\_growboth, 460
  - jbox\_get\_growy, 461
  - jbox\_get\_hidden, 461
  - jbox\_get\_hint, 461
  - jbox\_get\_hintstring, 461
  - jbox\_get\_id, 462
  - jbox\_get\_ignoreclick, 462
  - jbox\_get\_maxclass, 462
  - jbox\_get\_nextobject, 462

- jbox\_get\_nogrow, [463](#)
- jbox\_get\_object, [463](#)
- jbox\_get\_outline, [463](#)
- jbox\_get\_patcher, [463](#)
- jbox\_get\_patching\_position, [464](#)
- jbox\_get\_patching\_rect, [464](#)
- jbox\_get\_patching\_size, [464](#)
- jbox\_get\_presentation, [464](#)
- jbox\_get\_presentation\_position, [465](#)
- jbox\_get\_presentation\_rect, [465](#)
- jbox\_get\_presentation\_size, [465](#)
- jbox\_get\_prevobject, [465](#)
- jbox\_get\_rect\_for\_sym, [466](#)
- jbox\_get\_rect\_for\_view, [466](#)
- jbox\_get\_textfield, [466](#)
- jbox\_get\_varname, [466](#)
- jbox\_new, [467](#)
- JBOX\_NOINSPECTFIRSTIN, [458](#)
- jbox\_notify, [467](#)
- jbox\_ready, [467](#)
- jbox\_redraw, [467](#)
- jbox\_set\_annotation, [468](#)
- jbox\_set\_background, [468](#)
- jbox\_set\_color, [468](#)
- jbox\_set\_fontname, [468](#)
- jbox\_set\_fontsize, [469](#)
- jbox\_set\_hidden, [469](#)
- jbox\_set\_hint, [469](#)
- jbox\_set\_hintstring, [469](#)
- jbox\_set\_ignoreclick, [470](#)
- jbox\_set\_outline, [470](#)
- jbox\_set\_patching\_position, [470](#)
- jbox\_set\_patching\_rect, [470](#)
- jbox\_set\_patching\_size, [471](#)
- jbox\_set\_position, [471](#)
- jbox\_set\_presentation, [471](#)
- jbox\_set\_presentation\_position, [471](#)
- jbox\_set\_presentation\_rect, [472](#)
- jbox\_set\_presentation\_size, [472](#)
- jbox\_set\_rect, [472](#)
- jbox\_set\_rect\_for\_sym, [472](#)
- jbox\_set\_rect\_for\_view, [473](#)
- jbox\_set\_size, [473](#)
- jbox\_set\_varname, [473](#)
- JBOX\_FONTFACE\_BOLD
  - jbox, [458](#)
- JBOX\_FONTFACE\_BOLDITALIC
  - jbox, [458](#)
- JBOX\_FONTFACE\_ITALIC
  - jbox, [458](#)
- JBOX\_FONTFACE\_REGULAR
  - jbox, [458](#)
- jbox\_end\_layer
  - boxlayer, [589](#)
- jbox\_free
  - jbox, [458](#)
- jbox\_get\_annotation
  - jbox, [458](#)
- jbox\_get\_background
  - jbox, [459](#)
- jbox\_get\_canhilite
  - jbox, [459](#)
- jbox\_get\_color
  - jbox, [459](#)
- jbox\_get\_drawfirstin
  - jbox, [459](#)
- jbox\_get\_drawinlast
  - jbox, [460](#)
- jbox\_get\_font\_slant
  - jfont, [558](#)
- jbox\_get\_font\_weight
  - jfont, [559](#)
- jbox\_get\_fontname
  - jbox, [460](#)
- jbox\_get\_fontsize
  - jbox, [460](#)
- jbox\_get\_growboth
  - jbox, [460](#)
- jbox\_get\_growy
  - jbox, [461](#)
- jbox\_get\_hidden
  - jbox, [461](#)
- jbox\_get\_hint
  - jbox, [461](#)
- jbox\_get\_hintstring
  - jbox, [461](#)
- jbox\_get\_id
  - jbox, [462](#)
- jbox\_get\_ignoreclick
  - jbox, [462](#)
- jbox\_get\_maxclass
  - jbox, [462](#)
- jbox\_get\_nextobject
  - jbox, [462](#)
- jbox\_get\_nogrow
  - jbox, [463](#)
- jbox\_get\_object
  - jbox, [463](#)
- jbox\_get\_outline
  - jbox, [463](#)
- jbox\_get\_patcher
  - jbox, [463](#)
- jbox\_get\_patching\_position
  - jbox, [464](#)
- jbox\_get\_patching\_rect
  - jbox, [464](#)
- jbox\_get\_patching\_size
  - jbox, [464](#)

- jbox\_get\_presentation
  - jbox, [464](#)
- jbox\_get\_presentation\_position
  - jbox, [465](#)
- jbox\_get\_presentation\_rect
  - jbox, [465](#)
- jbox\_get\_presentation\_size
  - jbox, [465](#)
- jbox\_get\_prevobject
  - jbox, [465](#)
- jbox\_get\_rect\_for\_sym
  - jbox, [466](#)
- jbox\_get\_rect\_for\_view
  - jbox, [466](#)
- jbox\_get\_textfield
  - jbox, [466](#)
- jbox\_get\_varname
  - jbox, [466](#)
- jbox\_invalidate\_layer
  - boxlayer, [589](#)
- jbox\_new
  - jbox, [467](#)
- JBOX\_NOINSPECTFIRSTIN
  - jbox, [458](#)
- jbox\_notify
  - jbox, [467](#)
- jbox\_paint\_layer
  - boxlayer, [589](#)
- jbox\_ready
  - jbox, [467](#)
- jbox\_redraw
  - jbox, [467](#)
- jbox\_set\_annotation
  - jbox, [468](#)
- jbox\_set\_background
  - jbox, [468](#)
- jbox\_set\_color
  - jbox, [468](#)
- jbox\_set\_fontname
  - jbox, [468](#)
- jbox\_set\_fontsize
  - jbox, [469](#)
- jbox\_set\_hidden
  - jbox, [469](#)
- jbox\_set\_hint
  - jbox, [469](#)
- jbox\_set\_hintstring
  - jbox, [469](#)
- jbox\_set\_ignoreclick
  - jbox, [470](#)
- jbox\_set\_outline
  - jbox, [470](#)
- jbox\_set\_patching\_position
  - jbox, [470](#)
- jbox\_set\_patching\_rect
  - jbox, [470](#)
- jbox\_set\_patching\_size
  - jbox, [471](#)
- jbox\_set\_position
  - jbox, [471](#)
- jbox\_set\_presentation
  - jbox, [471](#)
- jbox\_set\_presentation\_position
  - jbox, [471](#)
- jbox\_set\_presentation\_rect
  - jbox, [472](#)
- jbox\_set\_presentation\_size
  - jbox, [472](#)
- jbox\_set\_rect
  - jbox, [472](#)
- jbox\_set\_rect\_for\_sym
  - jbox, [472](#)
- jbox\_set\_rect\_for\_view
  - jbox, [473](#)
- jbox\_set\_size
  - jbox, [473](#)
- jbox\_set\_varname
  - jbox, [473](#)
- jbox\_start\_layer
  - boxlayer, [589](#)
- jdataview
  - jdataview\_getclient, [592](#)
  - jdataview\_new, [592](#)
  - jdataview\_setclient, [592](#)
- jdataview\_getclient
  - jdataview, [592](#)
- jdataview\_new
  - jdataview, [592](#)
- jdataview\_setclient
  - jdataview, [592](#)
- JFont, [557](#)
- jfont
  - jbox\_get\_font\_slant, [558](#)
  - jbox\_get\_font\_weight, [559](#)
  - jfont\_create, [559](#)
  - jfont\_destroy, [559](#)
  - jfont\_extents, [559](#)
  - jfont\_getfontlist, [559](#)
  - jfont\_reference, [560](#)
  - jfont\_set\_font\_size, [560](#)
  - jfont\_set\_underline, [560](#)
  - jfont\_text\_measure, [560](#)
  - jfont\_text\_measure\_wrapped, [561](#)
  - JGRAPHICS\_FONT\_SLANT\_ITALIC, [558](#)
  - JGRAPHICS\_FONT\_SLANT\_NORMAL, [558](#)
  - JGRAPHICS\_FONT\_WEIGHT\_BOLD, [558](#)



- JGRAPHICS\_FONT\_WEIGHT\_NORMAL,  
558
  - t\_jgraphics\_font\_slant, 558
  - t\_jgraphics\_font\_weight, 558
- jfont\_create
  - jfont, 559
- jfont\_destroy
  - jfont, 559
- jfont\_extents
  - jfont, 559
- jfont\_getfontlist
  - jfont, 559
- jfont\_reference
  - jfont, 560
- jfont\_set\_font\_size
  - jfont, 560
- jfont\_set\_underline
  - jfont, 560
- jfont\_text\_measure
  - jfont, 560
- jfont\_text\_measure\_wrapped
  - jfont, 561
- JGraphics, 529
  - jgraphics
    - JGRAPHICS\_FILEFORMAT\_JPEG, 534
    - JGRAPHICS\_FILEFORMAT\_PNG, 534
    - JGRAPHICS\_FORMAT\_A8, 535
    - JGRAPHICS\_FORMAT\_ARGB32, 535
    - JGRAPHICS\_FORMAT\_RGB24, 535
    - JGRAPHICS\_TEXT\_JUSTIFICATION\_-  
BOTTOM, 535
    - JGRAPHICS\_TEXT\_JUSTIFICATION\_-  
CENTERED, 535
    - JGRAPHICS\_TEXT\_JUSTIFICATION\_-  
HCENTERED, 535
    - JGRAPHICS\_TEXT\_JUSTIFICATION\_-  
HJUSTIFIED, 535
    - JGRAPHICS\_TEXT\_JUSTIFICATION\_-  
LEFT, 535
    - JGRAPHICS\_TEXT\_JUSTIFICATION\_-  
RIGHT, 535
    - JGRAPHICS\_TEXT\_JUSTIFICATION\_TOP,  
535
    - JGRAPHICS\_TEXT\_JUSTIFICATION\_-  
VCENTERED, 535
    - JGRAPHICS\_2PI, 534
    - JGRAPHICS\_3PIOVER2, 534
    - jgraphics\_append\_path, 535
    - jgraphics\_arc, 535
    - jgraphics\_arc\_negative, 536
    - jgraphics\_close\_path, 536
    - jgraphics\_copy\_path, 536
    - jgraphics\_curve\_to, 536
    - jgraphics\_destroy, 536
    - jgraphics\_device\_to\_user, 537
    - jgraphics\_ellipse, 537
    - jgraphics\_font\_extents, 537
    - jgraphics\_get\_current\_point, 537
    - jgraphics\_getfiletypes, 537
    - jgraphics\_line\_to, 538
    - jgraphics\_move\_to, 538
    - jgraphics\_new\_path, 539
    - jgraphics\_oval, 539
    - jgraphics\_ovalarc, 539
    - jgraphics\_path\_destroy, 539
    - jgraphics\_path\_roundcorners, 539
    - JGRAPHICS\_PI, 534
    - JGRAPHICS\_PIOVER2, 534
    - jgraphics\_position\_one\_rect\_near\_another\_-  
rect\_but\_keep\_inside\_a\_third\_rect, 540
    - jgraphics\_rectangle, 540
    - jgraphics\_rectangle\_rounded, 540
    - jgraphics\_rectcontainsrect, 540
    - jgraphics\_rectintersectsrect, 541
    - jgraphics\_reference, 541
    - jgraphics\_rel\_curve\_to, 541
    - jgraphics\_rel\_line\_to, 541
    - jgraphics\_rel\_move\_to, 542
    - jgraphics\_round, 542
    - jgraphics\_select\_font\_face, 542
    - jgraphics\_select\_jfont, 542
    - jgraphics\_set\_font\_size, 542
    - jgraphics\_set\_underline, 543
    - jgraphics\_show\_text, 543
    - jgraphics\_system\_-  
canantialiastexttotransparentbg, 543
    - jgraphics\_text\_measure, 543
    - jgraphics\_text\_measure\_wrapped, 543
    - jgraphics\_user\_to\_device, 544
    - t\_jgraphics\_fileformat, 534
    - t\_jgraphics\_format, 534
    - t\_jgraphics\_text\_justification, 535
- JGraphics Matrix Transformations, 562
- JGRAPHICS\_FILEFORMAT\_JPEG
  - jgraphics, 534
- JGRAPHICS\_FILEFORMAT\_PNG
  - jgraphics, 534
- JGRAPHICS\_FONT\_SLANT\_ITALIC
  - jfont, 558
- JGRAPHICS\_FONT\_SLANT\_NORMAL
  - jfont, 558
- JGRAPHICS\_FONT\_WEIGHT\_BOLD
  - jfont, 558
- JGRAPHICS\_FONT\_WEIGHT\_NORMAL
  - jfont, 558
- JGRAPHICS\_FORMAT\_A8
  - jgraphics, 535
- JGRAPHICS\_FORMAT\_ARGB32

- jgraphics, 535
- JGRAPHICS\_FORMAT\_RGB24
  - jgraphics, 535
- JGRAPHICS\_TEXT\_JUSTIFICATION\_BOTTOM
  - jgraphics, 535
- JGRAPHICS\_TEXT\_JUSTIFICATION\_CENTERED
  - jgraphics, 535
- JGRAPHICS\_TEXT\_JUSTIFICATION\_HCENTERED
  - jgraphics, 535
- JGRAPHICS\_TEXT\_JUSTIFICATION\_HJUSTIFIED
  - jgraphics, 535
- JGRAPHICS\_TEXT\_JUSTIFICATION\_LEFT
  - jgraphics, 535
- JGRAPHICS\_TEXT\_JUSTIFICATION\_RIGHT
  - jgraphics, 535
- JGRAPHICS\_TEXT\_JUSTIFICATION\_TOP
  - jgraphics, 535
- JGRAPHICS\_TEXT\_JUSTIFICATION\_VCENTERED
  - jgraphics, 535
- JGRAPHICS\_TEXTLAYOUT\_NOWRAP
  - textlayout, 581
- JGRAPHICS\_TEXTLAYOUT\_USEELLIPSIS
  - textlayout, 581
- JGRAPHICS\_2PI
  - jgraphics, 534
- JGRAPHICS\_3PIOVER2
  - jgraphics, 534
- jgraphics\_append\_path
  - jgraphics, 535
- jgraphics\_arc
  - jgraphics, 535
- jgraphics\_arc\_negative
  - jgraphics, 536
- jgraphics\_close\_path
  - jgraphics, 536
- jgraphics\_copy\_path
  - jgraphics, 536
- jgraphics\_create
  - jsurface, 547
- jgraphics\_curve\_to
  - jgraphics, 536
- jgraphics\_destroy
  - jgraphics, 536
- jgraphics\_device\_to\_user
  - jgraphics, 537
- jgraphics\_ellipse
  - jgraphics, 537
- jgraphics\_font\_extents
  - jgraphics, 537
- jgraphics\_get\_current\_point
  - jgraphics, 537
- jgraphics\_get\_resource\_data
  - jsurface, 547
- jgraphics\_getfiletypes
  - jgraphics, 537
- jgraphics\_image\_surface\_clear
  - jsurface, 547
- jgraphics\_image\_surface\_create
  - jsurface, 547
- jgraphics\_image\_surface\_create\_for\_data
  - jsurface, 548
- jgraphics\_image\_surface\_create\_from\_file
  - jsurface, 548
- jgraphics\_image\_surface\_create\_from\_filedata
  - jsurface, 548
- jgraphics\_image\_surface\_create\_from\_resource
  - jsurface, 549
- jgraphics\_image\_surface\_create\_referenced
  - jsurface, 549
- jgraphics\_image\_surface\_draw
  - jsurface, 550
- jgraphics\_image\_surface\_draw\_fast
  - jsurface, 550
- jgraphics\_image\_surface\_get\_height
  - jsurface, 550
- jgraphics\_image\_surface\_get\_pixel
  - jsurface, 550
- jgraphics\_image\_surface\_get\_width
  - jsurface, 551
- jgraphics\_image\_surface\_lockpixels
  - jsurface, 551
- jgraphics\_image\_surface\_lockpixels\_readonly
  - jsurface, 551
- jgraphics\_image\_surface\_scroll
  - jsurface, 552
- jgraphics\_image\_surface\_set\_pixel
  - jsurface, 552
- jgraphics\_image\_surface\_unlockpixels
  - jsurface, 552
- jgraphics\_image\_surface\_unlockpixels\_readonly
  - jsurface, 552
- jgraphics\_image\_surface\_writepng
  - jsurface, 553
- jgraphics\_line\_to
  - jgraphics, 538
- jgraphics\_matrix\_init
  - jmatrix, 563
- jgraphics\_matrix\_init\_identity
  - jmatrix, 563
- jgraphics\_matrix\_init\_rotate
  - jmatrix, 563
- jgraphics\_matrix\_init\_scale
  - jmatrix, 563
- jgraphics\_matrix\_init\_translate

- jmatrix, [564](#)
- jgraphics\_matrix\_invert
  - jmatrix, [564](#)
- jgraphics\_matrix\_multiply
  - jmatrix, [564](#)
- jgraphics\_matrix\_rotate
  - jmatrix, [564](#)
- jgraphics\_matrix\_scale
  - jmatrix, [564](#)
- jgraphics\_matrix\_transform\_point
  - jmatrix, [565](#)
- jgraphics\_matrix\_translate
  - jmatrix, [565](#)
- jgraphics\_move\_to
  - jgraphics, [538](#)
- jgraphics\_new\_path
  - jgraphics, [539](#)
- jgraphics\_oval
  - jgraphics, [539](#)
- jgraphics\_ovalarc
  - jgraphics, [539](#)
- jgraphics\_path\_destroy
  - jgraphics, [539](#)
- jgraphics\_path\_roundcorners
  - jgraphics, [539](#)
- JGRAPHICS\_PI
  - jgraphics, [534](#)
- JGRAPHICS\_PIOVER2
  - jgraphics, [534](#)
- jgraphics\_position\_one\_rect\_near\_another\_rect\_  
but\_keep\_inside\_a\_third\_rect
  - jgraphics, [540](#)
- jgraphics\_rectangle
  - jgraphics, [540](#)
- jgraphics\_rectangle\_rounded
  - jgraphics, [540](#)
- jgraphics\_rectcontainsrect
  - jgraphics, [540](#)
- jgraphics\_rectintersectsrect
  - jgraphics, [541](#)
- jgraphics\_reference
  - jgraphics, [541](#)
- jgraphics\_rel\_curve\_to
  - jgraphics, [541](#)
- jgraphics\_rel\_line\_to
  - jgraphics, [541](#)
- jgraphics\_rel\_move\_to
  - jgraphics, [542](#)
- jgraphics\_round
  - jgraphics, [542](#)
- jgraphics\_select\_font\_face
  - jgraphics, [542](#)
- jgraphics\_select\_jfont
  - jgraphics, [542](#)
- jgraphics\_set\_font\_size
  - jgraphics, [542](#)
- jgraphics\_set\_underline
  - jgraphics, [543](#)
- jgraphics\_show\_text
  - jgraphics, [543](#)
- jgraphics\_surface\_destroy
  - jsurface, [553](#)
- jgraphics\_surface\_reference
  - jsurface, [553](#)
- jgraphics\_system\_canantialias\_text\_to\_transparent\_bg
  - jgraphics, [543](#)
- jgraphics\_text\_measure
  - jgraphics, [543](#)
- jgraphics\_text\_measure\_wrapped
  - jgraphics, [543](#)
- jgraphics\_user\_to\_device
  - jgraphics, [544](#)
- jgraphics\_write\_image\_surface\_to\_file\_data
  - jsurface, [553](#)
- jkeyboard\_getcurrentmodifiers
  - jmouse, [397](#)
- jkeyboard\_getcurrentmodifiers\_realtime
  - jmouse, [397](#)
- jmatrix
  - jgraphics\_matrix\_init, [563](#)
  - jgraphics\_matrix\_init\_identity, [563](#)
  - jgraphics\_matrix\_init\_rotate, [563](#)
  - jgraphics\_matrix\_init\_scale, [563](#)
  - jgraphics\_matrix\_init\_translate, [564](#)
  - jgraphics\_matrix\_invert, [564](#)
  - jgraphics\_matrix\_multiply, [564](#)
  - jgraphics\_matrix\_rotate, [564](#)
  - jgraphics\_matrix\_scale, [564](#)
  - jgraphics\_matrix\_transform\_point, [565](#)
  - jgraphics\_matrix\_translate, [565](#)
- jmonitor
  - jmonitor\_getdisplayrect, [392](#)
  - jmonitor\_getdisplayrect\_for\_all\_displays, [392](#)
  - jmonitor\_getdisplayrect\_for\_point, [392](#)
  - jmonitor\_getnumdisplays, [393](#)
- jmonitor\_getdisplayrect
  - jmonitor, [392](#)
- jmonitor\_getdisplayrect\_for\_all\_displays
  - jmonitor, [392](#)
- jmonitor\_getdisplayrect\_for\_point
  - jmonitor, [392](#)
- jmonitor\_getnumdisplays
  - jmonitor, [393](#)
- jmouse
  - eAltKey, [397](#)
  - eAutoRepeat, [397](#)
  - eCapsLock, [397](#)
  - eCommandKey, [397](#)

- eControlKey, [397](#)
- eLeftButton, [397](#)
- eMiddleButton, [397](#)
- ePopupMenu, [397](#)
- eRightButton, [397](#)
- eShiftKey, [397](#)
- jkeyboard\_getcurrentmodifiers, [397](#)
- jkeyboard\_getcurrentmodifiers\_realtime, [397](#)
- JMOUSE\_CURSOR\_ARROW, [396](#)
- JMOUSE\_CURSOR\_COPYING, [396](#)
- JMOUSE\_CURSOR\_CROSSHAIR, [396](#)
- JMOUSE\_CURSOR\_DRAGGINGHAND, [396](#)
- JMOUSE\_CURSOR\_IBEAM, [396](#)
- JMOUSE\_CURSOR\_NONE, [396](#)
- JMOUSE\_CURSOR\_POINTINGHAND, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_-  
BOTTOMEDGE, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_-  
BOTTOMLEFTCORNER, [397](#)
- JMOUSE\_CURSOR\_RESIZE\_-  
BOTTOMRIGHTCORNER, [397](#)
- JMOUSE\_CURSOR\_RESIZE\_FOURWAY, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_LEFTEDGE, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_LEFTRIGHT, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_-  
RIGHTEDGE, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_TOPEDGE, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_-  
TOPLEFTCORNER, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_-  
TOPRIGHTCORNER, [397](#)
- JMOUSE\_CURSOR\_RESIZE\_UPDOWN, [396](#)
- JMOUSE\_CURSOR\_WAIT, [396](#)
- jmouse\_getposition\_global, [397](#)
- jmouse\_setcursor, [398](#)
- jmouse\_setposition\_box, [398](#)
- jmouse\_setposition\_global, [398](#)
- jmouse\_setposition\_view, [398](#)
- t\_jmouse\_cursortype, [396](#)
- t\_modifiers, [397](#)
- JMOUSE\_CURSOR\_ARROW  
jmouse, [396](#)
- JMOUSE\_CURSOR\_COPYING  
jmouse, [396](#)
- JMOUSE\_CURSOR\_CROSSHAIR  
jmouse, [396](#)
- JMOUSE\_CURSOR\_DRAGGINGHAND  
jmouse, [396](#)
- JMOUSE\_CURSOR\_IBEAM  
jmouse, [396](#)
- JMOUSE\_CURSOR\_NONE  
jmouse, [396](#)
- JMOUSE\_CURSOR\_POINTINGHAND  
jmouse, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_BOTTOMEDGE  
jmouse, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_-  
BOTTOMLEFTCORNER  
jmouse, [397](#)
- JMOUSE\_CURSOR\_RESIZE\_-  
BOTTOMRIGHTCORNER  
jmouse, [397](#)
- JMOUSE\_CURSOR\_RESIZE\_FOURWAY  
jmouse, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_LEFTEDGE  
jmouse, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_LEFTRIGHT  
jmouse, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_RIGHTEDGE  
jmouse, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_TOPEDGE  
jmouse, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_-  
TOPLEFTCORNER  
jmouse, [396](#)
- JMOUSE\_CURSOR\_RESIZE\_-  
TOPRIGHTCORNER  
jmouse, [397](#)
- JMOUSE\_CURSOR\_RESIZE\_UPDOWN  
jmouse, [396](#)
- JMOUSE\_CURSOR\_WAIT  
jmouse, [396](#)
- jmouse\_getposition\_global  
jmouse, [397](#)
- jmouse\_setcursor  
jmouse, [398](#)
- jmouse\_setposition\_box  
jmouse, [398](#)
- jmouse\_setposition\_global  
jmouse, [398](#)
- jmouse\_setposition\_view  
jmouse, [398](#)
- jpatcher, [439](#)
- jpatcher\_deleteobj, [441](#)
- jpatcher\_get\_bgcolor, [442](#)
- jpatcher\_get\_bghidden, [442](#)
- jpatcher\_get\_bglocked, [442](#)
- jpatcher\_get\_box, [442](#)
- jpatcher\_get\_count, [442](#)
- jpatcher\_get\_currentfileversion, [443](#)
- jpatcher\_get\_default\_fontface, [443](#)
- jpatcher\_get\_default\_fontname, [443](#)

jpatcher\_get\_default\_fontsize, 443  
jpatcher\_get\_defrect, 443  
jpatcher\_get\_dirty, 444  
jpatcher\_get\_editing\_bgcolor, 444  
jpatcher\_get\_fghidden, 444  
jpatcher\_get\_filename, 444  
jpatcher\_get\_filepath, 445  
jpatcher\_get\_fileversion, 445  
jpatcher\_get\_firstline, 445  
jpatcher\_get\_firstobject, 445  
jpatcher\_get\_firstview, 446  
jpatcher\_get\_gridsize, 446  
jpatcher\_get\_lastobject, 446  
jpatcher\_get\_name, 447  
jpatcher\_get\_parentpatcher, 447  
jpatcher\_get\_presentation, 447  
jpatcher\_get\_rect, 447  
jpatcher\_get\_title, 447  
jpatcher\_get\_toppatcher, 448  
jpatcher\_is\_patcher, 448  
jpatcher\_set\_bgcolor, 448  
jpatcher\_set\_bghidden, 448  
jpatcher\_set\_bglocked, 449  
jpatcher\_set\_defrect, 449  
jpatcher\_set\_dirty, 449  
jpatcher\_set\_editing\_bgcolor, 449  
jpatcher\_set\_fghidden, 450  
jpatcher\_set\_gridsize, 450  
jpatcher\_set\_locked, 450  
jpatcher\_set\_presentation, 450  
jpatcher\_set\_rect, 451  
jpatcher\_set\_title, 451  
jpatcher\_uniqueboxname, 451  
jpatcher\_deleteobj  
    jpatcher, 441  
jpatcher\_get\_bgcolor  
    jpatcher, 442  
jpatcher\_get\_bghidden  
    jpatcher, 442  
jpatcher\_get\_bglocked  
    jpatcher, 442  
jpatcher\_get\_box  
    jpatcher, 442  
jpatcher\_get\_count  
    jpatcher, 442  
jpatcher\_get\_currentfileversion  
    jpatcher, 443  
jpatcher\_get\_default\_fontface  
    jpatcher, 443  
jpatcher\_get\_default\_fontname  
    jpatcher, 443  
jpatcher\_get\_default\_fontsize  
    jpatcher, 443  
jpatcher\_get\_defrect  
    jpatcher, 443  
jpatcher\_get\_dirty  
    jpatcher, 444  
jpatcher\_get\_editing\_bgcolor  
    jpatcher, 444  
jpatcher\_get\_fghidden  
    jpatcher, 444  
jpatcher\_get\_filename  
    jpatcher, 444  
jpatcher\_get\_filepath  
    jpatcher, 445  
jpatcher\_get\_fileversion  
    jpatcher, 445  
jpatcher\_get\_firstline  
    jpatcher, 445  
jpatcher\_get\_firstobject  
    jpatcher, 445  
jpatcher\_get\_firstview  
    jpatcher, 446  
jpatcher\_get\_gridsize  
    jpatcher, 446  
jpatcher\_get\_lastobject  
    jpatcher, 446  
jpatcher\_get\_name  
    jpatcher, 447  
jpatcher\_get\_parentpatcher  
    jpatcher, 447  
jpatcher\_get\_presentation  
    jpatcher, 447  
jpatcher\_get\_rect  
    jpatcher, 447  
jpatcher\_get\_title  
    jpatcher, 447  
jpatcher\_get\_toppatcher  
    jpatcher, 448  
jpatcher\_is\_patcher  
    jpatcher, 448  
jpatcher\_set\_bgcolor  
    jpatcher, 448  
jpatcher\_set\_bghidden  
    jpatcher, 448  
jpatcher\_set\_bglocked  
    jpatcher, 449  
jpatcher\_set\_defrect  
    jpatcher, 449  
jpatcher\_set\_dirty  
    jpatcher, 449  
jpatcher\_set\_editing\_bgcolor  
    jpatcher, 449  
jpatcher\_set\_fghidden  
    jpatcher, 450  
jpatcher\_set\_gridsize  
    jpatcher, 450  
jpatcher\_set\_locked

- jpatcher, 450
- jpatcher\_set\_presentation
  - jpatcher, 450
- jpatcher\_set\_rect
  - jpatcher, 451
- jpatcher\_set\_title
  - jpatcher, 451
- jpatcher\_uniqueboxname
  - jpatcher, 451
- jpatcherview, 478
  - patcherview\_findpatcherview, 479
  - patcherview\_get\_jgraphics, 479
  - patcherview\_get\_locked, 479
  - patcherview\_get\_nextview, 479
  - patcherview\_get\_patcher, 480
  - patcherview\_get\_presentation, 480
  - patcherview\_get\_rect, 480
  - patcherview\_get\_visible, 480
  - patcherview\_get\_zoomfactor, 481
  - patcherview\_set\_locked, 481
  - patcherview\_set\_presentation, 481
  - patcherview\_set\_rect, 481
  - patcherview\_set\_visible, 482
  - patcherview\_set\_zoomfactor, 482
- jpatchline, 474
  - jpatchline\_get\_box1, 475
  - jpatchline\_get\_box2, 475
  - jpatchline\_get\_color, 475
  - jpatchline\_get\_endpoint, 475
  - jpatchline\_get\_hidden, 476
  - jpatchline\_get\_inletnum, 476
  - jpatchline\_get\_nextline, 476
  - jpatchline\_get\_nummidpoints, 476
  - jpatchline\_get\_outletnum, 476
  - jpatchline\_get\_startpoint, 477
  - jpatchline\_set\_color, 477
  - jpatchline\_set\_hidden, 477
- jpatchline\_get\_box1
  - jpatchline, 475
- jpatchline\_get\_box2
  - jpatchline, 475
- jpatchline\_get\_color
  - jpatchline, 475
- jpatchline\_get\_endpoint
  - jpatchline, 475
- jpatchline\_get\_hidden
  - jpatchline, 476
- jpatchline\_get\_inletnum
  - jpatchline, 476
- jpatchline\_get\_nextline
  - jpatchline, 476
- jpatchline\_get\_nummidpoints
  - jpatchline, 476
- jpatchline\_get\_outletnum
  - jpatchline, 476
- jpatchline\_get\_startpoint
  - jpatchline, 477
- jpatchline\_set\_color
  - jpatchline, 477
- jpatchline\_set\_hidden
  - jpatchline, 477
- JPattern, 566
- jpopupmenu
  - jpopupmenu\_additem, 585
  - jpopupmenu\_addseparator, 585
  - jpopupmenu\_addsubmenu, 585
  - jpopupmenu\_clear, 585
  - jpopupmenu\_create, 585
  - jpopupmenu\_destroy, 586
  - jpopupmenu\_popup, 586
  - jpopupmenu\_popup\_abovebox, 586
  - jpopupmenu\_popup\_nearbox, 586
  - jpopupmenu\_setcolors, 587
  - jpopupmenu\_setfont, 587
- jpopupmenu\_additem
  - jpopupmenu, 585
- jpopupmenu\_addseparator
  - jpopupmenu, 585
- jpopupmenu\_addsubmenu
  - jpopupmenu, 585
- jpopupmenu\_clear
  - jpopupmenu, 585
- jpopupmenu\_create
  - jpopupmenu, 585
- jpopupmenu\_destroy
  - jpopupmenu, 586
- jpopupmenu\_popup
  - jpopupmenu, 586
- jpopupmenu\_popup\_abovebox
  - jpopupmenu, 586
- jpopupmenu\_popup\_nearbox
  - jpopupmenu, 586
- jpopupmenu\_setcolors
  - jpopupmenu, 587
- jpopupmenu\_setfont
  - jpopupmenu, 587
- jrgba\_attr\_get
  - color, 568
- jrgba\_attr\_set
  - color, 568
- jrgba\_compare
  - color, 568
- jrgba\_copy
  - color, 569
- jrgba\_set
  - color, 569
- jrgba\_to\_atoms
  - color, 569

- JSurface, 545
- jsurface
  - jgraphics\_create, 547
  - jgraphics\_get\_resource\_data, 547
  - jgraphics\_image\_surface\_clear, 547
  - jgraphics\_image\_surface\_create, 547
  - jgraphics\_image\_surface\_create\_for\_data, 548
  - jgraphics\_image\_surface\_create\_from\_file, 548
  - jgraphics\_image\_surface\_create\_from\_filedata, 548
  - jgraphics\_image\_surface\_create\_from\_resource, 549
  - jgraphics\_image\_surface\_create\_referenced, 549
  - jgraphics\_image\_surface\_draw, 550
  - jgraphics\_image\_surface\_draw\_fast, 550
  - jgraphics\_image\_surface\_get\_height, 550
  - jgraphics\_image\_surface\_get\_pixel, 550
  - jgraphics\_image\_surface\_get\_width, 551
  - jgraphics\_image\_surface\_lockpixels, 551
  - jgraphics\_image\_surface\_lockpixels\_readonly, 551
  - jgraphics\_image\_surface\_scroll, 552
  - jgraphics\_image\_surface\_set\_pixel, 552
  - jgraphics\_image\_surface\_unlockpixels, 552
  - jgraphics\_image\_surface\_unlockpixels\_readonly, 552
  - jgraphics\_image\_surface\_writepng, 553
  - jgraphics\_surface\_destroy, 553
  - jgraphics\_surface\_reference, 553
  - jgraphics\_write\_image\_surface\_to\_filedata, 553
- jsvg
  - jsvg\_create\_from\_file, 555
  - jsvg\_create\_from\_resource, 555
  - jsvg\_create\_from\_xmlstring, 556
  - jsvg\_destroy, 556
  - jsvg\_get\_size, 556
  - jsvg\_render, 556
- jsvg\_create\_from\_file
  - jsvg, 555
- jsvg\_create\_from\_resource
  - jsvg, 555
- jsvg\_create\_from\_xmlstring
  - jsvg, 556
- jsvg\_destroy
  - jsvg, 556
- jsvg\_get\_size
  - jsvg, 556
- jsvg\_render
  - jsvg, 556
- jtextlayout\_create
  - textlayout, 581
- jtextlayout\_destroy
  - textlayout, 581
- jtextlayout\_draw
  - textlayout, 581
- jtextlayout\_getchar
  - textlayout, 581
- jtextlayout\_getcharbox
  - textlayout, 582
- jtextlayout\_getnumchars
  - textlayout, 582
- jtextlayout\_measure
  - textlayout, 582
- jtextlayout\_set
  - textlayout, 582
- jtextlayout\_settextcolor
  - textlayout, 583
- jtextlayout\_withbgcolor
  - textlayout, 583
- jwind
  - jwind\_getactive, 394
  - jwind\_getat, 394
  - jwind\_getcount, 394
- jwind\_getactive
  - jwind, 394
- jwind\_getat
  - jwind, 394
- jwind\_getcount
  - jwind, 394
- Linked List, 271
- linklist
  - linklist\_append, 273
  - linklist\_chuck, 274
  - linklist\_chuckindex, 274
  - linklist\_chuckobject, 274
  - linklist\_clear, 275
  - linklist\_deleteindex, 275
  - linklist\_findall, 275
  - linklist\_findfirst, 276
  - linklist\_flags, 276
  - linklist\_funall, 277
  - linklist\_funall\_break, 277
  - linklist\_funindex, 278
  - linklist\_getflags, 278
  - linklist\_getindex, 278
  - linklist\_getsize, 278
  - linklist\_insert\_sorted, 279
  - linklist\_insertafterobjptr, 279
  - linklist\_insertbeforeobjptr, 279
  - linklist\_insertindex, 279
  - linklist\_last, 280
  - linklist\_makearray, 280
  - linklist\_match, 280
  - linklist\_methodall, 281



- linklist\_methodindex, 281
- linklist\_moveafterobjptr, 281
- linklist\_movebeforeobjptr, 282
- linklist\_new, 282
- linklist\_next, 282
- linklist\_objptr2index, 282
- linklist\_prev, 283
- linklist\_readonly, 283
- linklist\_reverse, 283
- linklist\_rotate, 283
- linklist\_shuffle, 283
- linklist\_sort, 284
- linklist\_substitute, 284
- linklist\_swap, 284
- linklist\_append
  - linklist, 273
- linklist\_chuck
  - linklist, 274
- linklist\_chuckindex
  - linklist, 274
- linklist\_chuckobject
  - linklist, 274
- linklist\_clear
  - linklist, 275
- linklist\_deleteindex
  - linklist, 275
- linklist\_findall
  - linklist, 275
- linklist\_findfirst
  - linklist, 276
- linklist\_flags
  - linklist, 276
- linklist\_funall
  - linklist, 277
- linklist\_funall\_break
  - linklist, 277
- linklist\_funindex
  - linklist, 278
- linklist\_getflags
  - linklist, 278
- linklist\_getindex
  - linklist, 278
- linklist\_getsize
  - linklist, 278
- linklist\_insert\_sorted
  - linklist, 279
- linklist\_insertafterobjptr
  - linklist, 279
- linklist\_insertbeforeobjptr
  - linklist, 279
- linklist\_insertindex
  - linklist, 279
- linklist\_last
  - linklist, 280
- linklist\_makearray
  - linklist, 280
- linklist\_match
  - linklist, 280
- linklist\_methodall
  - linklist, 281
- linklist\_methodindex
  - linklist, 281
- linklist\_moveafterobjptr
  - linklist, 281
- linklist\_movebeforeobjptr
  - linklist, 282
- linklist\_new
  - linklist, 282
- linklist\_next
  - linklist, 282
- linklist\_objptr2index
  - linklist, 282
- linklist\_prev
  - linklist, 283
- linklist\_readonly
  - linklist, 283
- linklist\_reverse
  - linklist, 283
- linklist\_rotate
  - linklist, 283
- linklist\_shuffle
  - linklist, 283
- linklist\_sort
  - linklist, 284
- linklist\_substitute
  - linklist, 284
- linklist\_swap
  - linklist, 284
- listout
  - inout, 206
- Loading Max Files, 389
- loading\_max\_files
  - fileload, 389
  - intload, 389
  - readtohandle, 390
  - stringload, 390
- locatefile
  - files, 333
- locatefile\_extended
  - files, 333
- locatefiletype
  - files, 334
- MAX
  - misc, 359
- MAX\_ERR\_DUPLICATE
  - misc, 360
- MAX\_ERR\_GENERIC



- [misc](#), [360](#)
- [MAX\\_ERR\\_INVALID\\_PTR](#)
  - [misc](#), [360](#)
- [MAX\\_ERR\\_NONE](#)
  - [misc](#), [360](#)
- [MAX\\_ERR\\_OUT\\_OF\\_MEM](#)
  - [misc](#), [360](#)
- [MAX\\_FILENAME\\_CHARS](#)
  - [files](#), [330](#)
- [MAXARG](#)
  - [obj](#), [417](#)
- [maxversion](#)
  - [misc](#), [362](#)
- [memory](#)
  - [disposhandle](#), [349](#)
  - [freebytes](#), [349](#)
  - [freebytes16](#), [349](#)
  - [getbytes](#), [350](#)
  - [getbytes16](#), [350](#)
  - [growhandle](#), [350](#)
  - [MM\\_UNIFIED](#), [349](#)
  - [newhandle](#), [350](#)
  - [system\\_copyptr](#), [351](#)
  - [system\\_freehandle](#), [351](#)
  - [system\\_freeptr](#), [351](#)
  - [system\\_handlesize](#), [351](#)
  - [system\\_lockhandle](#), [351](#)
  - [system\\_newhandle](#), [352](#)
  - [system\\_newhandleclear](#), [352](#)
  - [system\\_newptr](#), [352](#)
  - [system\\_newptrclear](#), [352](#)
  - [system\\_nullterminatehandle](#), [353](#)
  - [system\\_ptrandhand](#), [353](#)
  - [system\\_ptrbeforehand](#), [353](#)
  - [system\\_ptrsize](#), [353](#)
  - [system\\_resizehandle](#), [354](#)
  - [system\\_resizeptr](#), [354](#)
  - [system\\_resizeptrclear](#), [354](#)
- [Memory Management](#), [347](#)
- [MIN](#)
  - [misc](#), [359](#)
- [misc](#)
  - [aaCancel](#), [360](#)
  - [aaNo](#), [360](#)
  - [aaYes](#), [360](#)
  - [BEGIN\\_USING\\_C\\_LINKAGE](#), [358](#)
  - [calcoffset](#), [358](#)
  - [CLIP](#), [358](#)
  - [e\\_max\\_errorcodes](#), [360](#)
  - [e\\_max\\_wind\\_advise\\_result](#), [360](#)
  - [error\\_subscribe](#), [360](#)
  - [error\\_sym](#), [360](#)
  - [error\\_unsubscribe](#), [361](#)
  - [globalsymbol\\_bind](#), [361](#)
  - [globalsymbol\\_dereference](#), [361](#)
  - [globalsymbol\\_reference](#), [361](#)
  - [globalsymbol\\_unbind](#), [362](#)
  - [InRange](#), [359](#)
  - [MAX](#), [359](#)
  - [MAX\\_ERR\\_DUPLICATE](#), [360](#)
  - [MAX\\_ERR\\_GENERIC](#), [360](#)
  - [MAX\\_ERR\\_INVALID\\_PTR](#), [360](#)
  - [MAX\\_ERR\\_NONE](#), [360](#)
  - [MAX\\_ERR\\_OUT\\_OF\\_MEM](#), [360](#)
  - [maxversion](#), [362](#)
  - [MIN](#), [359](#)
  - [object\\_obex\\_quickref](#), [363](#)
  - [post\\_sym](#), [363](#)
  - [quittask\\_install](#), [363](#)
  - [quittask\\_remove](#), [363](#)
  - [snprintf\\_zero](#), [363](#)
  - [strncat\\_zero](#), [364](#)
  - [strncpy\\_zero](#), [364](#)
  - [symbol\\_unique](#), [364](#)
  - [symbolarray\\_sort](#), [364](#)
  - [wind\\_advise](#), [365](#)
  - [wind\\_setcursor](#), [365](#)
- [Miscellaneous](#), [355](#)
- [MM\\_UNIFIED](#)
  - [memory](#), [349](#)
- [Monitors and Displays](#), [392](#)
- [Mouse and Keyboard](#), [395](#)
- [MSP](#), [402](#)
- [msp](#)
  - [class\\_dspinit](#), [405](#)
  - [class\\_dspinitbox](#), [405](#)
  - [class\\_dspinitjbox](#), [406](#)
  - [dsp\\_add](#), [406](#)
  - [dsp\\_addv](#), [406](#)
  - [PI](#), [404](#)
  - [PIOVERTWO](#), [404](#)
  - [SYS\\_MAXBLKSIZE](#), [405](#)
  - [SYS\\_MAXSIGS](#), [405](#)
  - [sys\\_getblksize](#), [406](#)
  - [sys\\_getdspstate](#), [407](#)
  - [sys\\_getmaxblksize](#), [407](#)
  - [sys\\_getsr](#), [407](#)
  - [t\\_float](#), [404](#)
  - [t\\_int](#), [405](#)
  - [t\\_perfroutine](#), [405](#)
  - [t\\_sample](#), [405](#)
  - [TWOPI](#), [404](#)
  - [z\\_dsp\\_free](#), [407](#)
  - [z\\_dsp\\_setup](#), [407](#)
- [mutex](#)
  - [systhread\\_mutex\\_free](#), [525](#)
  - [systhread\\_mutex\\_lock](#), [525](#)
  - [systhread\\_mutex\\_new](#), [526](#)

- systhread\_mutex\_newlock, [526](#)
  - systhread\_mutex\_trylock, [526](#)
  - systhread\_mutex\_unlock, [527](#)
- Mutexes, [525](#)
- newhandle
  - memory, [350](#)
- newinstance
  - class\_old, [200](#)
- newobject
  - class\_old, [201](#)
- newobject\_fromdictionary
  - obj, [417](#)
- newobject\_sprintf
  - obj, [418](#)
- OBEX\_UTIL\_ATOM\_GETTEXT\_COMMA\_-  
DELIM
  - atom, [299](#)
- OBEX\_UTIL\_ATOM\_GETTEXT\_DEFAULT
  - atom, [298](#)
- OBEX\_UTIL\_ATOM\_GETTEXT\_FORCE\_-  
ZEROS
  - atom, [299](#)
- OBEX\_UTIL\_ATOM\_GETTEXT\_NUM\_HI\_RES
  - atom, [299](#)
- OBEX\_UTIL\_ATOM\_GETTEXT\_SYM\_-  
FORCE\_QUOTE
  - atom, [299](#)
- OBEX\_UTIL\_ATOM\_GETTEXT\_SYM\_NO\_-  
QUOTE
  - atom, [299](#)
- OBEX\_UTIL\_ATOM\_GETTEXT\_TRUNCATE\_-  
ZEROS
  - atom, [298](#)
- obj
  - classname\_openhelp, [417](#)
  - classname\_openquery, [417](#)
  - classname\_openrefpage, [417](#)
  - MAXARG, [417](#)
  - newobject\_fromdictionary, [417](#)
  - newobject\_sprintf, [418](#)
  - object\_alloc, [419](#)
  - object\_attach, [419](#)
  - object\_attach\_byptr, [419](#)
  - object\_attach\_byptr\_register, [420](#)
  - object\_class, [420](#)
  - object\_classname, [421](#)
  - object\_classname\_compare, [421](#)
  - object\_detach, [421](#)
  - object\_detach\_byptr, [421](#)
  - object\_dictionaryarg, [422](#)
  - object\_findregistered, [422](#)
  - object\_findregisteredbyptr, [422](#)
  - object\_free, [423](#)
  - object\_getmethod, [423](#)
  - object\_getvalueof, [423](#)
  - object\_method, [424](#)
  - object\_method\_char, [425](#)
  - object\_method\_char\_array, [425](#)
  - object\_method\_double, [425](#)
  - object\_method\_double\_array, [426](#)
  - object\_method\_float, [426](#)
  - object\_method\_float\_array, [427](#)
  - object\_method\_format, [427](#)
  - object\_method\_long, [427](#)
  - object\_method\_long\_array, [428](#)
  - object\_method\_obj, [428](#)
  - object\_method\_obj\_array, [429](#)
  - object\_method\_parse, [429](#)
  - object\_method\_sym, [429](#)
  - object\_method\_sym\_array, [430](#)
  - object\_method\_typed, [430](#)
  - object\_method\_typedfun, [431](#)
  - object\_new, [431](#)
  - object\_new\_typed, [431](#)
  - object\_notify, [432](#)
  - object\_obex\_dumpout, [432](#)
  - object\_obex\_lookup, [433](#)
  - object\_obex\_store, [433](#)
  - object\_openhelp, [434](#)
  - object\_openquery, [434](#)
  - object\_openrefpage, [434](#)
  - object\_register, [434](#)
  - object\_setvalueof, [435](#)
  - object\_unregister, [435](#)
- OBJ\_FLAG\_DATA
  - datastore, [212](#)
- OBJ\_FLAG\_MEMORY
  - datastore, [212](#)
- OBJ\_FLAG\_OBJ
  - datastore, [212](#)
- OBJ\_FLAG\_REF
  - datastore, [212](#)
- OBJ\_FLAG\_SILENT
  - datastore, [212](#)
- OBJ\_ATTR\_ATOM
  - attr, [152](#)
- OBJ\_ATTR\_ATOM\_ARRAY
  - attr, [153](#)
- OBJ\_ATTR\_CHAR
  - attr, [153](#)
- OBJ\_ATTR\_CHAR\_ARRAY
  - attr, [153](#)
- OBJ\_ATTR\_DEFAULT
  - attr, [153](#)
- OBJ\_ATTR\_DEFAULT\_SAVE
  - attr, [154](#)

- OBJ\_ATTR\_DOUBLE
  - [attr, 154](#)
- OBJ\_ATTR\_DOUBLE\_ARRAY
  - [attr, 154](#)
- OBJ\_ATTR\_FLOAT
  - [attr, 155](#)
- OBJ\_ATTR\_FLOAT\_ARRAY
  - [attr, 155](#)
- OBJ\_ATTR\_LONG
  - [attr, 155](#)
- OBJ\_ATTR\_LONG\_ARRAY
  - [attr, 156](#)
- OBJ\_ATTR\_OBJ
  - [attr, 156](#)
- OBJ\_ATTR\_OBJ\_ARRAY
  - [attr, 156](#)
- OBJ\_ATTR\_SAVE
  - [attr, 156](#)
- OBJ\_ATTR\_SYM
  - [attr, 157](#)
- OBJ\_ATTR\_SYM\_ARRAY
  - [attr, 157](#)
- [object\\_addattr](#)
  - [attr, 177](#)
- [object\\_alloc](#)
  - [obj, 419](#)
- [object\\_attach](#)
  - [obj, 419](#)
- [object\\_attach\\_byptr](#)
  - [obj, 419](#)
- [object\\_attach\\_byptr\\_register](#)
  - [obj, 420](#)
- [object\\_attr\\_get](#)
  - [attr, 177](#)
- [object\\_attr\\_get\\_rect](#)
  - [attr, 178](#)
- [object\\_attr\\_getchar\\_array](#)
  - [attr, 178](#)
- [object\\_attr\\_getcolor](#)
  - [attr, 178](#)
- [object\\_attr\\_getdouble\\_array](#)
  - [attr, 179](#)
- [object\\_attr\\_getdump](#)
  - [attr, 179](#)
- [object\\_attr\\_getfloat](#)
  - [attr, 179](#)
- [object\\_attr\\_getfloat\\_array](#)
  - [attr, 180](#)
- [object\\_attr\\_getjrgba](#)
  - [attr, 180](#)
- [object\\_attr\\_getlong](#)
  - [attr, 180](#)
- [object\\_attr\\_getlong\\_array](#)
  - [attr, 181](#)
- [object\\_attr\\_getpt](#)
  - [attr, 181](#)
- [object\\_attr\\_getsize](#)
  - [attr, 181](#)
- [object\\_attr\\_getsym](#)
  - [attr, 182](#)
- [object\\_attr\\_getsym\\_array](#)
  - [attr, 182](#)
- [object\\_attr\\_method](#)
  - [attr, 182](#)
- [object\\_attr\\_set\\_rect](#)
  - [attr, 183](#)
- [object\\_attr\\_setchar\\_array](#)
  - [attr, 183](#)
- [object\\_attr\\_setcolor](#)
  - [attr, 183](#)
- [object\\_attr\\_setdouble\\_array](#)
  - [attr, 184](#)
- [object\\_attr\\_setfloat](#)
  - [attr, 184](#)
- [object\\_attr\\_setfloat\\_array](#)
  - [attr, 184](#)
- [object\\_attr\\_setjrgba](#)
  - [attr, 185](#)
- [object\\_attr\\_setlong](#)
  - [attr, 185](#)
- [object\\_attr\\_setlong\\_array](#)
  - [attr, 185](#)
- [object\\_attr\\_setparse](#)
  - [attr, 186](#)
- [object\\_attr\\_setpt](#)
  - [attr, 186](#)
- [object\\_attr\\_setsize](#)
  - [attr, 186](#)
- [object\\_attr\\_setsym](#)
  - [attr, 187](#)
- [object\\_attr\\_setsym\\_array](#)
  - [attr, 187](#)
- [object\\_attr\\_setvalueof](#)
  - [attr, 187](#)
- [object\\_attr\\_usercanget](#)
  - [attr, 188](#)
- [object\\_attr\\_usercanset](#)
  - [attr, 188](#)
- [object\\_chuckattr](#)
  - [attr, 188](#)
- [object\\_class](#)
  - [obj, 420](#)
- [object\\_classname](#)
  - [obj, 421](#)
- [object\\_classname\\_compare](#)
  - [obj, 421](#)
- [object\\_deleteattr](#)
  - [attr, 189](#)

- object\_detach
  - obj, [421](#)
- object\_detach\_byptr
  - obj, [421](#)
- object\_dictionaryarg
  - obj, [422](#)
- object\_error
  - console, [367](#)
- object\_error\_obtrusive
  - console, [368](#)
- object\_findregistered
  - obj, [422](#)
- object\_findregisteredbyptr
  - obj, [422](#)
- object\_free
  - obj, [423](#)
- object\_getmethod
  - obj, [423](#)
- object\_getvalueof
  - obj, [423](#)
- object\_method
  - obj, [424](#)
- object\_method\_char
  - obj, [425](#)
- object\_method\_char\_array
  - obj, [425](#)
- object\_method\_double
  - obj, [425](#)
- object\_method\_double\_array
  - obj, [426](#)
- object\_method\_float
  - obj, [426](#)
- object\_method\_float\_array
  - obj, [427](#)
- object\_method\_format
  - obj, [427](#)
- object\_method\_long
  - obj, [427](#)
- object\_method\_long\_array
  - obj, [428](#)
- object\_method\_obj
  - obj, [428](#)
- object\_method\_obj\_array
  - obj, [429](#)
- object\_method\_parse
  - obj, [429](#)
- object\_method\_sym
  - obj, [429](#)
- object\_method\_sym\_array
  - obj, [430](#)
- object\_method\_typed
  - obj, [430](#)
- object\_method\_typedfun
  - obj, [431](#)
- object\_new
  - obj, [431](#)
- object\_new\_parse
  - attr, [189](#)
- object\_new\_typed
  - obj, [431](#)
- object\_notify
  - obj, [432](#)
- object\_obex\_dumpout
  - obj, [432](#)
- object\_obex\_lookup
  - obj, [433](#)
- object\_obex\_quickref
  - misc, [363](#)
- object\_obex\_store
  - obj, [433](#)
- object\_openhelp
  - obj, [434](#)
- object\_openquery
  - obj, [434](#)
- object\_openrefpage
  - obj, [434](#)
- object\_post
  - console, [368](#)
- object\_register
  - obj, [434](#)
- object\_setvalueof
  - obj, [435](#)
- object\_unregister
  - obj, [435](#)
- object\_warn
  - console, [368](#)
- Objects, [413](#)
- Old-Style Classes, [197](#)
- open\_dialog
  - files, [335](#)
- open\_promptset
  - files, [335](#)
- ouchstring
  - console, [369](#)
- outlet\_anything
  - inout, [206](#)
- outlet\_bang
  - inout, [207](#)
- outlet\_float
  - inout, [207](#)
- outlet\_int
  - inout, [207](#)
- outlet\_list
  - inout, [207](#)
- outlet\_new
  - inout, [208](#)
- Patcher, [437](#)

- patcher
  - PI\_DEEP, [438](#)
  - PI\_REQUIREFIRSTIN, [438](#)
  - PI\_WANTBOX, [438](#)
  - t\_box, [438](#)
  - t\_patcher, [438](#)
- patcherview\_findpatcherview
  - jpatcherview, [479](#)
- patcherview\_get\_jgraphics
  - jpatcherview, [479](#)
- patcherview\_get\_locked
  - jpatcherview, [479](#)
- patcherview\_get\_nextview
  - jpatcherview, [479](#)
- patcherview\_get\_patcher
  - jpatcherview, [480](#)
- patcherview\_get\_presentation
  - jpatcherview, [480](#)
- patcherview\_get\_rect
  - jpatcherview, [480](#)
- patcherview\_get\_visible
  - jpatcherview, [480](#)
- patcherview\_get\_zoomfactor
  - jpatcherview, [481](#)
- patcherview\_set\_locked
  - jpatcherview, [481](#)
- patcherview\_set\_presentation
  - jpatcherview, [481](#)
- patcherview\_set\_rect
  - jpatcherview, [481](#)
- patcherview\_set\_visible
  - jpatcherview, [482](#)
- patcherview\_set\_zoomfactor
  - jpatcherview, [482](#)
- PATH\_FILEINFO\_ALIAS
  - files, [330](#)
- PATH\_FILEINFO\_FOLDER
  - files, [330](#)
- PATH\_FILEINFO\_PACKAGE
  - files, [330](#)
- PATH\_FOLDER\_SNIFF
  - files, [331](#)
- PATH\_READ\_PERM
  - files, [331](#)
- PATH\_REPORTPACKAGEASFOLDER
  - files, [331](#)
- PATH\_RW\_PERM
  - files, [331](#)
- PATH\_STYLE\_COLON
  - files, [331](#)
- PATH\_STYLE\_MAX
  - files, [331](#)
- PATH\_STYLE\_NATIVE
  - files, [331](#)
- PATH\_STYLE\_NATIVE\_WIN
  - files, [331](#)
- PATH\_STYLE\_SLASH
  - files, [331](#)
- PATH\_TYPE\_ABSOLUTE
  - files, [332](#)
- PATH\_TYPE\_BOOT
  - files, [332](#)
- PATH\_TYPE\_C74
  - files, [332](#)
- PATH\_TYPE\_IGNORE
  - files, [332](#)
- PATH\_TYPE\_PATH
  - files, [332](#)
- PATH\_TYPE\_RELATIVE
  - files, [332](#)
- PATH\_WRITE\_PERM
  - files, [331](#)
- path\_closefolder
  - files, [335](#)
- path\_createsysfile
  - files, [336](#)
- path\_fileinfo
  - files, [336](#)
- path\_foldernextfile
  - files, [336](#)
- path\_frompathname
  - files, [337](#)
- path\_getapppath
  - files, [337](#)
- path\_getdefault
  - files, [337](#)
- path\_getfilemoddate
  - files, [337](#)
- path\_getmoddate
  - files, [337](#)
- path\_nameconform
  - files, [338](#)
- path\_openfolder
  - files, [338](#)
- path\_opensysfile
  - files, [338](#)
- path\_resolvefile
  - files, [339](#)
- path\_setdefault
  - files, [339](#)
- path\_topathname
  - files, [339](#)
- path\_topotentialname
  - files, [340](#)
- PFFT, [411](#)
- PI
  - msp, [404](#)
- PI\_DEEP

- patcher, [438](#)
- PI\_REQUIREFIRSTIN
  - patcher, [438](#)
- PI\_WANTBOX
  - patcher, [438](#)
- PIOVERTWO
  - msp, [404](#)
- Poly, [412](#)
- Popup Menus, [584](#)
- post
  - console, [369](#)
- post\_sym
  - misc, [363](#)
- postargs
  - atom, [313](#)
- postatom
  - console, [370](#)
- postdictionary
  - dictionary, [255](#)
- preset\_int
  - presets, [385](#)
- preset\_set
  - presets, [385](#)
- preset\_store
  - presets, [385](#)
- Presets, [384](#)
- presets
  - preset\_int, [385](#)
  - preset\_set, [385](#)
  - preset\_store, [385](#)
- proxy\_getinlet
  - inout, [208](#)
- proxy\_new
  - inout, [209](#)
- qelem\_free
  - qelems, [494](#)
- qelem\_front
  - qelems, [494](#)
- qelem\_new
  - qelems, [494](#)
- qelem\_set
  - qelems, [494](#)
- qelem\_unset
  - qelems, [495](#)
- Qelems, [493](#)
- qelems
  - qelem\_free, [494](#)
  - qelem\_front, [494](#)
  - qelem\_new, [494](#)
  - qelem\_set, [494](#)
  - qelem\_unset, [495](#)
- Quick Map, [286](#)
- quickmap
  - quickmap\_add, [286](#)
  - quickmap\_drop, [287](#)
  - quickmap\_lookup\_key1, [287](#)
  - quickmap\_lookup\_key2, [287](#)
  - quickmap\_new, [287](#)
  - quickmap\_readonly, [288](#)
- quickmap\_add
  - quickmap, [286](#)
- quickmap\_drop
  - quickmap, [287](#)
- quickmap\_lookup\_key1
  - quickmap, [287](#)
- quickmap\_lookup\_key2
  - quickmap, [287](#)
- quickmap\_new
  - quickmap, [287](#)
- quickmap\_readonly
  - quickmap, [288](#)
- quittask\_install
  - misc, [363](#)
- quittask\_remove
  - misc, [363](#)
- readatom
  - binbuf, [320](#)
- readtohandle
  - loading\_max\_files, [390](#)
- saveas\_dialog
  - files, [340](#)
- saveas\_promptset
  - files, [340](#)
- saveasdialog\_extended
  - files, [341](#)
- Scalable Vector Graphics, [555](#)
- schedule
  - threading, [518](#)
- schedule\_delay
  - threading, [519](#)
- scheduler\_gettime
  - clocks, [489](#)
- scheduler\_new
  - clocks, [490](#)
- scheduler\_run
  - clocks, [490](#)
- scheduler\_set
  - clocks, [490](#)
- scheduler\_settime
  - clocks, [490](#)
- serialno
  - evnum, [387](#)
- setclock\_delay
  - clocks, [491](#)
- setclock\_fdelay

- clocks, [491](#)
- setclock\_gettime
  - clocks, [491](#)
- setclock\_gettime
  - clocks, [492](#)
- setclock\_unset
  - clocks, [492](#)
- setup
  - class\_old, [201](#)
- snprintf\_zero
  - misc, [363](#)
- STATIC\_ATTR\_ATOM
  - attr, [157](#)
- STATIC\_ATTR\_ATOM\_ARRAY
  - attr, [158](#)
- STATIC\_ATTR\_CHAR
  - attr, [158](#)
- STATIC\_ATTR\_CHAR\_ARRAY
  - attr, [158](#)
- STATIC\_ATTR\_DOUBLE
  - attr, [158](#)
- STATIC\_ATTR\_DOUBLE\_ARRAY
  - attr, [159](#)
- STATIC\_ATTR\_FLOAT
  - attr, [159](#)
- STATIC\_ATTR\_FLOAT\_ARRAY
  - attr, [159](#)
- STATIC\_ATTR\_LONG
  - attr, [160](#)
- STATIC\_ATTR\_LONG\_ARRAY
  - attr, [160](#)
- STATIC\_ATTR\_OBJ
  - attr, [160](#)
- STATIC\_ATTR\_OBJ\_ARRAY
  - attr, [160](#)
- STATIC\_ATTR\_SYM
  - attr, [161](#)
- STATIC\_ATTR\_SYM\_ARRAY
  - attr, [161](#)
- string
  - string\_getptr, [289](#)
  - string\_new, [289](#)
- String Object, [289](#)
- string\_getptr
  - string, [289](#)
- string\_new
  - string, [289](#)
- stringload
  - loading\_max\_files, [390](#)
- strncat\_zero
  - misc, [364](#)
- strncpy\_zero
  - misc, [364](#)
- STRUCT\_ATTR\_ATOM
  - attr, [161](#)
- STRUCT\_ATTR\_ATOM\_ARRAY
  - attr, [162](#)
- STRUCT\_ATTR\_ATOM\_VARSIZE
  - attr, [162](#)
- STRUCT\_ATTR\_CHAR
  - attr, [162](#)
- STRUCT\_ATTR\_CHAR\_ARRAY
  - attr, [163](#)
- STRUCT\_ATTR\_CHAR\_VARSIZE
  - attr, [163](#)
- STRUCT\_ATTR\_DOUBLE
  - attr, [163](#)
- STRUCT\_ATTR\_DOUBLE\_ARRAY
  - attr, [164](#)
- STRUCT\_ATTR\_DOUBLE\_VARSIZE
  - attr, [164](#)
- STRUCT\_ATTR\_FLOAT
  - attr, [164](#)
- STRUCT\_ATTR\_FLOAT\_ARRAY
  - attr, [165](#)
- STRUCT\_ATTR\_FLOAT\_VARSIZE
  - attr, [165](#)
- STRUCT\_ATTR\_LONG
  - attr, [165](#)
- STRUCT\_ATTR\_LONG\_ARRAY
  - attr, [166](#)
- STRUCT\_ATTR\_LONG\_VARSIZE
  - attr, [166](#)
- STRUCT\_ATTR\_OBJ
  - attr, [166](#)
- STRUCT\_ATTR\_OBJ\_ARRAY
  - attr, [167](#)
- STRUCT\_ATTR\_OBJ\_VARSIZE
  - attr, [167](#)
- STRUCT\_ATTR\_SYM
  - attr, [167](#)
- STRUCT\_ATTR\_SYM\_ARRAY
  - attr, [168](#)
- STRUCT\_ATTR\_SYM\_VARSIZE
  - attr, [168](#)
- symbol
  - gensym, [323](#)
- Symbol Object, [291](#)
- symbol\_unique
  - misc, [364](#)
- symbolarray\_sort
  - misc, [364](#)
- Symbols, [322](#)
- symobject
  - symobject\_linklist\_match, [291](#)
  - symobject\_new, [292](#)
- symobject\_linklist\_match
  - symobject, [291](#)

- symobject\_new
  - symobject, [292](#)
- SYS\_MAXBLKSIZE
  - msp, [405](#)
- SYS\_MAXSIGS
  - msp, [405](#)
- sys\_getblksize
  - msp, [406](#)
- sys\_getdspstate
  - msp, [407](#)
- sys\_getmaxblksize
  - msp, [407](#)
- sys\_getsr
  - msp, [407](#)
- SYSDATEFORMAT\_FLAGS\_LONG
  - system, [497](#)
- SYSDATEFORMAT\_FLAGS\_MEDIUM
  - system, [497](#)
- SYSDATEFORMAT\_FLAGS\_SHORT
  - system, [497](#)
- sysdateformat\_formatdatetime
  - system, [497](#)
- sysdateformat\_strftimeodatetime
  - system, [497](#)
- SYSFILE\_ATMARK
  - files, [332](#)
- SYSFILE\_FROMLEOF
  - files, [332](#)
- SYSFILE\_FROMMARK
  - files, [332](#)
- SYSFILE\_FROMSTART
  - files, [332](#)
- sysfile\_close
  - files, [342](#)
- sysfile\_geteof
  - files, [342](#)
- sysfile\_getpos
  - files, [342](#)
- sysfile\_openhandle
  - files, [343](#)
- sysfile\_openptrsize
  - files, [343](#)
- sysfile\_read
  - files, [343](#)
- sysfile\_readtextfile
  - files, [344](#)
- sysfile\_readtohandle
  - files, [344](#)
- sysfile\_readtoptr
  - files, [344](#)
- sysfile\_seteof
  - files, [345](#)
- sysfile\_setpos
  - files, [345](#)
- sysfile\_spoolcopy
  - files, [345](#)
- sysfile\_write
  - files, [345](#)
- sysfile\_writetextfile
  - files, [346](#)
- sysmem\_copyptr
  - memory, [351](#)
- sysmem\_freehandle
  - memory, [351](#)
- sysmem\_freeptr
  - memory, [351](#)
- sysmem\_handlesize
  - memory, [351](#)
- sysmem\_lockhandle
  - memory, [351](#)
- sysmem\_newhandle
  - memory, [352](#)
- sysmem\_newhandleclear
  - memory, [352](#)
- sysmem\_newptr
  - memory, [352](#)
- sysmem\_newptrclear
  - memory, [352](#)
- sysmem\_nullterminatehandle
  - memory, [353](#)
- sysmem\_ptrandhand
  - memory, [353](#)
- sysmem\_ptrbeforehand
  - memory, [353](#)
- sysmem\_ptrsize
  - memory, [353](#)
- sysmem\_resizehandle
  - memory, [354](#)
- sysmem\_resizeptr
  - memory, [354](#)
- sysmem\_resizeptrclear
  - memory, [354](#)
- SYSTHREAD\_MUTEX\_ERRORCHECK
  - threading, [516](#)
- SYSTHREAD\_MUTEX\_NORMAL
  - threading, [516](#)
- SYSTHREAD\_MUTEX\_RECURSIVE
  - threading, [516](#)
- systhread\_create
  - threading, [519](#)
- systhread\_exit
  - threading, [520](#)
- systhread\_ismainthread
  - threading, [520](#)
- systhread\_istimerthread
  - threading, [520](#)
- systhread\_join
  - threading, [520](#)



- systhread\_mutex\_free
  - mutex, [525](#)
- systhread\_mutex\_lock
  - mutex, [525](#)
- systhread\_mutex\_new
  - mutex, [526](#)
- systhread\_mutex\_newlock
  - mutex, [526](#)
- systhread\_mutex\_trylock
  - mutex, [526](#)
- systhread\_mutex\_unlock
  - mutex, [527](#)
- systhread\_self
  - threading, [521](#)
- systhread\_sleep
  - threading, [521](#)
- systhread\_terminate
  - threading, [521](#)
- systime
  - e\_max\_dateflags, [497](#)
  - SYSDATEFORMAT\_FLAGS\_LONG, [497](#)
  - SYSDATEFORMAT\_FLAGS\_MEDIUM, [497](#)
  - SYSDATEFORMAT\_FLAGS\_SHORT, [497](#)
  - sysdateformat\_formatdatetime, [497](#)
  - sysdateformat\_strftimetodatetime, [497](#)
  - systime\_datetime, [497](#)
  - systime\_datetoseconds, [498](#)
  - systime\_ms, [498](#)
  - systime\_seconds, [498](#)
  - systime\_secondstodate, [498](#)
  - systime\_ticks, [498](#)
- Systime API, [496](#)
- systime\_datetime
  - systime, [497](#)
- systime\_datetoseconds
  - systime, [498](#)
- systime\_ms
  - systime, [498](#)
- systime\_seconds
  - systime, [498](#)
- systime\_secondstodate
  - systime, [498](#)
- systime\_ticks
  - systime, [498](#)
- systimer\_gettime
  - clocks, [492](#)
- t\_atom, [598](#)
- t\_atomarray, [599](#)
- t\_atombuf, [600](#)
- t\_attr, [601](#)
- t\_box
  - patcher, [438](#)
- t\_buffer, [602](#)
- t\_celldesc, [605](#)
- t\_charset\_converter, [606](#)
- t\_class, [607](#)
- t\_cmpfn
  - datastore, [211](#)
- t\_database
  - database, [221](#)
- t\_datetime, [608](#)
- t\_db\_result
  - database, [221](#)
- t\_db\_view
  - database, [221](#)
- t\_dictionary, [609](#)
- t\_dictionary\_entry, [611](#)
- t\_expr, [612](#)
- t\_filehandle
  - files, [330](#)
- t\_fileinfo, [613](#)
- t\_float
  - msp, [404](#)
- t\_funbuff, [614](#)
- t\_hashtab, [616](#)
- t\_hashtab\_entry, [617](#)
- t\_indexmap, [618](#)
- t\_indexmap\_entry, [620](#)
- t\_int
  - msp, [405](#)
- t\_itm
  - time, [513](#)
- t\_jbox, [621](#)
- t\_jboxdrawparams, [622](#)
- t\_jcolumn, [623](#)
- t\_jdataview, [627](#)
- t\_jgraphics\_fileformat
  - jgraphics, [534](#)
- t\_jgraphics\_font\_extents, [631](#)
- t\_jgraphics\_font\_slant
  - jfont, [558](#)
- t\_jgraphics\_font\_weight
  - jfont, [558](#)
- t\_jgraphics\_format
  - jgraphics, [534](#)
- t\_jgraphics\_text\_justification
  - jgraphics, [535](#)
- t\_jgraphics\_textlayout\_flags
  - textlayout, [581](#)
- t\_jmatrix, [632](#)
- t\_jmouse\_cursortype
  - jmouse, [396](#)
- t\_jrgb, [633](#)
- t\_jrgba, [634](#)
- t\_linklist, [635](#)
- t\_llelem, [636](#)
- t\_max\_err

- datatypes, 294
- t\_messlist, 637
- t\_modifiers
  - jmouse, 397
- t\_object, 638
- t\_patcher
  - patcher, 438
- t\_path, 639
- t\_pathlink, 640
- t\_perfroutine
  - msp, 405
- t\_pfftpub, 641
- t\_privatesortrec, 643
- t\_pt, 645
- t\_pxjbox, 646
- t\_pxobject, 647
- t\_quickmap, 648
- t\_rect, 649
- t\_sample
  - msp, 405
- t\_signal, 650
- t\_size, 651
- t\_string, 652
- t\_symbol, 653
- t\_symobject, 654
- t\_timeobject
  - time, 513
- t\_tinyobject, 655
- t\_zll, 656
- Table Access, 381
- table\_dirty
  - tables, 381
- table\_get
  - tables, 381
- tables
  - table\_dirty, 381
  - table\_get, 381
- Text Editor Windows, 383
- TEXT\_ENCODING\_USE\_FILE
  - files, 332
- TEXT\_LB\_MAC
  - files, 332
- TEXT\_LB\_NATIVE
  - files, 332
- TEXT\_LB\_PC
  - files, 332
- TEXT\_LB\_UNIX
  - files, 332
- TEXT\_NULL\_TERMINATE
  - files, 332
- TextField, 570
- textfield
  - textfield\_get\_autoscroll, 572
  - textfield\_get\_bgcolor, 572
  - textfield\_get\_editonclick, 572
  - textfield\_get\_emptytext, 572
  - textfield\_get\_noactivate, 573
  - textfield\_get\_owner, 573
  - textfield\_get\_readonly, 573
  - textfield\_get\_selectallonedit, 573
  - textfield\_get\_textcolor, 573
  - textfield\_get\_textmargins, 574
  - textfield\_get\_underline, 574
  - textfield\_get\_useellipsis, 574
  - textfield\_get\_wantsreturn, 574
  - textfield\_get\_wantstab, 575
  - textfield\_get\_wordwrap, 575
  - textfield\_set\_autoscroll, 575
  - textfield\_set\_bgcolor, 575
  - textfield\_set\_editonclick, 576
  - textfield\_set\_emptytext, 576
  - textfield\_set\_noactivate, 576
  - textfield\_set\_readonly, 576
  - textfield\_set\_selectallonedit, 577
  - textfield\_set\_textcolor, 577
  - textfield\_set\_textmargins, 577
  - textfield\_set\_underline, 577
  - textfield\_set\_useellipsis, 578
  - textfield\_set\_wantsreturn, 578
  - textfield\_set\_wantstab, 578
  - textfield\_set\_wordwrap, 578
- textfield\_get\_autoscroll
  - textfield, 572
- textfield\_get\_bgcolor
  - textfield, 572
- textfield\_get\_editonclick
  - textfield, 572
- textfield\_get\_emptytext
  - textfield, 572
- textfield\_get\_noactivate
  - textfield, 573
- textfield\_get\_owner
  - textfield, 573
- textfield\_get\_readonly
  - textfield, 573
- textfield\_get\_selectallonedit
  - textfield, 573
- textfield\_get\_textcolor
  - textfield, 573
- textfield\_get\_textmargins
  - textfield, 574
- textfield\_get\_underline
  - textfield, 574
- textfield\_get\_useellipsis
  - textfield, 574
- textfield\_get\_wantsreturn
  - textfield, 574
- textfield\_get\_wantstab
  - textfield, 574

- textfield, 575
- textfield\_get\_wordwrap
  - textfield, 575
- textfield\_set\_autoscroll
  - textfield, 575
- textfield\_set\_bgcolor
  - textfield, 575
- textfield\_set\_editonclick
  - textfield, 576
- textfield\_set\_emptytext
  - textfield, 576
- textfield\_set\_noactivate
  - textfield, 576
- textfield\_set\_readonly
  - textfield, 576
- textfield\_set\_selectallonedit
  - textfield, 577
- textfield\_set\_textcolor
  - textfield, 577
- textfield\_set\_textmargins
  - textfield, 577
- textfield\_set\_underline
  - textfield, 577
- textfield\_set\_useellipsis
  - textfield, 578
- textfield\_set\_wantsreturn
  - textfield, 578
- textfield\_set\_wantstab
  - textfield, 578
- textfield\_set\_wordwrap
  - textfield, 578
- TextLayout, 580
- textlayout
  - JGRAPHICS\_TEXTLAYOUT\_NOWRAP, 581
  - JGRAPHICS\_TEXTLAYOUT\_-USEELLIPSIS, 581
  - jtextlayout\_create, 581
  - jtextlayout\_destroy, 581
  - jtextlayout\_draw, 581
  - jtextlayout\_getchar, 581
  - jtextlayout\_getcharbox, 582
  - jtextlayout\_getnumchars, 582
  - jtextlayout\_measure, 582
  - jtextlayout\_set, 582
  - jtextlayout\_settextcolor, 583
  - jtextlayout\_withbgcolor, 583
  - t\_jgraphics\_textlayout\_flags, 581
- threading
  - ATOMIC\_DECREMENT, 516
  - ATOMIC\_INCREMENT, 516
  - defer, 517
  - defer\_low, 517
  - e\_max\_systhread\_mutex\_flags, 516
  - isr, 518
  - schedule, 518
  - schedule\_delay, 519
  - SYSTHREAD\_MUTEX\_ERRORCHECK, 516
  - SYSTHREAD\_MUTEX\_NORMAL, 516
  - SYSTHREAD\_MUTEX\_RECURSIVE, 516
  - systhread\_create, 519
  - systhread\_exit, 520
  - systhread\_ismainthread, 520
  - systhread\_istimerthread, 520
  - systhread\_join, 520
  - systhread\_self, 521
  - systhread\_sleep, 521
  - systhread\_terminate, 521
- Threads, 514
- time
  - class\_time\_addattr, 504
  - itm\_barbeatunitstoticks, 504
  - itm\_dereference, 504
  - itm\_dump, 504
  - itm\_getglobal, 505
  - itm\_getname, 505
  - itm\_getnamed, 505
  - itm\_getresolution, 505
  - itm\_getstate, 505
  - itm\_getticks, 506
  - itm\_gettime, 506
  - itm\_gettimesignature, 506
  - itm\_isunitfixed, 506
  - itm\_mstosamps, 507
  - itm\_mstoticks, 507
  - itm\_pause, 507
  - itm\_reference, 507
  - itm\_resume, 508
  - itm\_sampstoms, 508
  - itm\_setresolution, 508
  - itm\_settimesignature, 508
  - itm\_tickstobarbeatunits, 508
  - itm\_tickstoms, 509
  - t\_itm, 513
  - t\_timeobject, 513
  - TIME\_FLAGS\_BBUSOURCE, 504
  - TIME\_FLAGS\_CHECKSCHEDULE, 503
  - TIME\_FLAGS\_EVENTLIST, 503
  - TIME\_FLAGS\_FIXED, 503
  - TIME\_FLAGS\_FIXEDONLY, 503
  - TIME\_FLAGS\_LISTENTICKS, 504
  - TIME\_FLAGS\_LOCATION, 503
  - TIME\_FLAGS\_LOOKAHEAD, 503
  - TIME\_FLAGS\_NOUNITS, 504
  - TIME\_FLAGS\_PERMANENT, 503
  - TIME\_FLAGS\_POSITIVE, 504
  - TIME\_FLAGS\_TICKSONLY, 503

- TIME\_FLAGS\_TRANSPORT, [503](#)
- TIME\_FLAGS\_USECLOCK, [503](#)
- TIME\_FLAGS\_USEQELEM, [503](#)
- time\_calcquantize, [509](#)
- time\_getitm, [509](#)
- time\_getms, [509](#)
- time\_getnamed, [510](#)
- time\_getphase, [510](#)
- time\_getticks, [510](#)
- time\_isfixedunit, [510](#)
- time\_listen, [511](#)
- time\_new, [511](#)
- time\_now, [511](#)
- time\_schedule, [511](#)
- time\_schedule\_limit, [512](#)
- time\_setclock, [512](#)
- time\_setvalue, [512](#)
- time\_stop, [512](#)
- time\_tick, [512](#)
- TIME\_FLAGS\_BBUSOURCE
  - time, [504](#)
- TIME\_FLAGS\_CHECKSCHEDULE
  - time, [503](#)
- TIME\_FLAGS\_EVENTLIST
  - time, [503](#)
- TIME\_FLAGS\_FIXED
  - time, [503](#)
- TIME\_FLAGS\_FIXEDONLY
  - time, [503](#)
- TIME\_FLAGS\_LISTENTICKS
  - time, [504](#)
- TIME\_FLAGS\_LOCATION
  - time, [503](#)
- TIME\_FLAGS\_LOOKAHEAD
  - time, [503](#)
- TIME\_FLAGS\_NOUNITS
  - time, [504](#)
- TIME\_FLAGS\_PERMANENT
  - time, [503](#)
- TIME\_FLAGS\_POSITIVE
  - time, [504](#)
- TIME\_FLAGS\_TICKSONLY
  - time, [503](#)
- TIME\_FLAGS\_TRANSPORT
  - time, [503](#)
- TIME\_FLAGS\_USECLOCK
  - time, [503](#)
- TIME\_FLAGS\_USEQELEM
  - time, [503](#)
- time\_calcquantize
  - time, [509](#)
- time\_getitm
  - time, [509](#)
- time\_getms
  - time, [509](#)
- time\_getnamed
  - time, [510](#)
- time\_getphase
  - time, [510](#)
- time\_getticks
  - time, [510](#)
- time\_isfixedunit
  - time, [510](#)
- time\_listen
  - time, [511](#)
- time\_new
  - time, [511](#)
- time\_now
  - time, [511](#)
- time\_schedule
  - time, [511](#)
- time\_schedule\_limit
  - time, [512](#)
- time\_setclock
  - time, [512](#)
- time\_setvalue
  - time, [512](#)
- time\_stop
  - time, [512](#)
- time\_tick
  - time, [512](#)
- Timing, [483](#)
- TWOPI
  - msp, [404](#)
- typedmess
  - class\_old, [202](#)
- Unicode, [593](#)
- unicode
  - charset\_convert, [594](#)
  - charset\_unicodetoutf8, [594](#)
  - charset\_utf8\_count, [595](#)
  - charset\_utf8\_offset, [595](#)
  - charset\_utf8tounicode, [595](#)
- User Interface, [528](#)
- wind\_advise
  - misc, [365](#)
- wind\_setcursor
  - misc, [365](#)
- Windows, [394](#)
- word, [657](#)
- z\_dsp\_free
  - msp, [407](#)
- z\_dsp\_setup
  - msp, [407](#)
- zgetfn
  - class\_old, [202](#)