

## **Notice**

Smart Technology makes no express or implied warranty with regard to the documentation offered. The documentation is made available solely on an 'as is' basis and the entire risk as to its quality and performance is with the user. Should the documentation and program material prove defective, the user (and not Smart Technology or any other party) shall bear the entire cost of all necessary correction and all incidental or consequential damages. Smart Technology shall not be liable for any incidental or consequential damages in connection with or arising out of the furnishing, use or performance of the documentation and programs.

## **FOREWORD**

This document contains the complete description of each hidden command residing in the Command's Library and some examples on how those commands could be used for non trivial tasks. The intent is that to make available to advanced users a considerable amount of general purpose routines which stay behind the scene during normal operation with the SmartROM. Most of the commands have a minimal error detection capability, thus use them with care. Error detecting feature is demanded to higher level code (first of all stack checking and argument type checking) so you get minimal overhead when executing them. A couple of routines have been designed to accomplish very general tasks without sacrificing speed or compactness of code. This could appeal those programmers who wish maximum flexibility and expandibility of code. Because sometimes could be difficult to foresee an application for certain routines, we have tried to give an idea of how many different things can be done with the same basic code. These suggestions are collected in the Notes section of each command.

Smart Technology obviously guarantees the stability of the code in terms of Input/Output conditions and Command identification numbers.

## Legenda

meta	means a meta-object with real counter.
meta <sub>h</sub>	means a meta-object with sysbin counter.
Comp	means a compound object: list, program or algebraic.
SymbMat or SymbMat(m,n)	means a list of lists containing Symbolics, reals, complex numbers, units. Sublists must be isotropic, that is each sublist must contain the same number of objects.
SymbMat(n)	means a square (n x n) SymbMat. Sublists must be isotropic, that is each sublist must contain the same number of objects.
ListMat(m,n)	means a list of lists (m x n). Sublists must be isotropic.
Symbolic	means a symbolic value (unit included) or a number (complex or real).
Obj	Any Object
TRUE	Means the internal definition of boolean value TRUE.
FALSE	Means the internal definition of boolean value FALSE.
Prg	A subroutine written either in system RPL or in user RPL.
<0> <pos>	A name or value surrounded by angle brackets is a system binary value.
<LID>	LID means <b>L</b> ibrary <b>I</b> Dentification number.

## Notes

Whenever the standard meaning is not explicitly specified, the term matrix means a list of list.

Whenever the contrary is not specified all the examples can be 'compiled' directly with the application program L->THREAD.

## Commands sorted by category

This section contains several tables organizing hidden commands in proper categories for easier referencing.

## String Utilities

Name	Page Number	Xlib #
center\$	23	244
find\$	117	156
lfpos	87	186
lines->	131	142
->lines	175	98
ltrim\$	48	225
lwc\$	49	224
mcentr\$	22	245
member\$	59	214
->msg\$	53	220
norm\$	51	222
null	148	125
pad\$	13	253
replace\$	154	119
revs	55	218
rowcol	158	115
rpt	159	114
rtrim\$	47	226
span\$	58	215
split\$	18	248
splith	168	105
trim\$	52	221
upc\$	50	223

## Binary string Utilities

addp	40	230
expbuf	24	243
null	148	125
popp	39	231
revb	54	219
rpt	159	114

### **System Binary Utilities**

log2	132	141
ord	38	232
revsys	70	203
todd	156	117

### **List Manipulation Utilities**

bind	82	191
ckl	83	190
ckl?	103	170
ckl2r	85	188
delcol	110	163
delrow	112	161
diff	27	240
findobj	120	153
getcol	73	200
idx?	127	146
inter	28	239
l2m	129	144
lget	130	143
lop1	133	140
lopn	134	139
lput	135	138
lvop	136	137
nget	67	206
npos	61	212
nput	66	207
null	148	125
putobj	65	208
red	26	241
replace	153	120
rpt	159	114
splitl	169	104
splito	17	249
union	29	238

### **General purpose Utilities**

#k	79	194
ckr	76	197
ckrol	77	196
getaddr	63	210
keywait	128	145
ncount	62	211
rptcmd	160	113

### **Program editing utilities**

findobj	120	153
#k->\$	78	195
nget	67	206
npos	61	212
nput	66	207
putobj	65	208
replace	153	120
search	166	107
splitl	169	104

### **Stack Manipulation Utilities**

c2m	102	171
hdrop	123	150
hdup	124	149
hshift	125	148
mark	137	136
mds	88	185
mus	91	182
rd	92	181
rdown	150	123
ru	93	180
rup	161	112
xlev	177	96
xlvs	174	99

### **Meta-object Manipulation Utilities**

bsearch	10	263
ckobj	84	189
copy	109	164
delete	111	162
ma2	71	202
maddl	9	264
metax	72	201
move	141	132
mrev	142	131
mtop	144	129
mtopn	145	128
ndupn	147	126
pkmeta	149	124
sort	12	254
sortany	163	110

## **Symbolic Math Utilities**

add	98	175
addcon	99	174
apply	68	205
best	165	108
cmpl	106	167
conform?	107	166
const	108	165
delcol	110	163
delrow	112	161
det2	119	154
det3	118	155
determ	113	160
dims	114	159
dot	157	116
equal?	115	158
factor	116	157
findrow	164	109
getcol	73	200
idn	126	147
mat->	138	135
->mat	176	97
msymb?	143	130
mult	146	127
reduc	155	118
square?	170	103
srccol	171	102
subt	172	101
trn	173	100
weight	167	106

## **Type Conversion Utilities**

->ext	69	204
->prolog	57	216
->torf	14	252
xlib->	56	217

## Graphics Utilities

box	33	236
fill	45	228
gaddr	121	152
gaddup	122	151
gop	34	235
line	35	234
linetypes	30	237
patterns	41	229
polygon	19	247
ppardef	21	246
rect	37	233
rpt	159	114
scan	60	213
scanp	25	242
vfill	46	227

## Object Manipulation Utilities

apply	68	205
chl?	103	170
chset?	104	169
chst?	105	168
dot	157	116
findobj	120	153
nget	67	206
npos	61	212
nput	66	207
null	148	125
putobj	65	208
replace	153	120
rpt	159	114
tifc	64	209

## Configuration

\$CONFIG





## Command Reference

**TOOLDIR Name:** madd1

**Type:** System RPL

**Xlib #** 264

**Input:**

Obj<sub>1</sub> Obj<sub>2</sub> ... Obj<sub>n</sub> <n> <pos> Obj

**Output:**

Obj<sub>1</sub> Obj<sub>2</sub> ...Obj Obj<sub>n</sub> <n+1>

**Purpose:** Inserts the input object above the position specified.

**Notes:** If pos is zero the object is inserted at the head of the meta-object. If pos is greater than n, the object is appended to the tail.

**TOOLDIR Name:** bsearch  
**Type:** System RPL  
**Xlib #** 263

**Input:**

Obj<sub>1</sub> Obj<sub>2</sub> ...      Obj<sub>n</sub>      <n>      Obj      Comparison

**Output:**

Obj<sub>1</sub> Obj<sub>2</sub> ...      Obj<sub>n</sub>      <n>      <pos>      TRUE

or

Obj<sub>1</sub> Obj<sub>2</sub> ...      Obj<sub>n</sub>      <n>      <where>      Obj      FALSE

**Purpose:**

Searches the input object in the meta-object provided the meta-object is in ascending order and returns the matching position or the best fitting place where it could be inserted.

**Notes:**

Comparison must be a procedure of the type "Greater than" taking two objects from the stack and returning a system boolean.

The value returned is suitable for use with maddl.

Null meta-objects are not allowed.

**TOOLDIR Name:** Various sort calls

**Type:** System RPL

**Xlib #** 255-262

**Input:**

Obj<sub>1</sub>      ...      Obj<sub>n</sub>      <n>      Any

**Output:**

Obj<sub>s1</sub>      ...      Obj<sub>sn</sub>      <n>

**Purpose:** Sort objects of specified type in ascending order using sortany.

**Notes:** Routines perform the sort on the following classes:

sortr	real numbers	
sortre	complex numbers	ordered by real part
sortu	units	
sortb	binary integers	
sortsb	system binaries	
sort\$	strings	
sortg	global names	ordered as strings
n/a	tagged objects	whose ancestor must be one of the above.

The first object is always dropped, because typically it comes from CKOBJ and has no more meaning.

**Uses:** sortany, sortm

**TOOLDIR Name:** sort

**Type:** System RPL

**Xlib #** 254

**Input:**

Obj<sub>1</sub>      ...      Obj<sub>n</sub>      <n>

**Output:**

Obj<sub>s1</sub>      ...      Obj<sub>sn</sub>      <n>

**Purpose:** Sorts objects in ascending order using default procedures.

**Notes:** Objects must fall in the following categories:

real numbers	
complex numbers	ordered by real part
units	
binary integers	
system binaries	
strings	
global names	ordered as strings
tagged objects	whose ancestor must be one of the above.

**Uses:** sortany, sortr, sortre, sort\$, sortsb, sortb, sortu, sortg, sortm

**TOOLDIR Name:** pad\$

**Type:** Program

**Xlib #** 253

**Input:**

3: String  
2: Pad\_String  
1: <length>

**Output:**

String

**Purpose:** Pads the input string with the string on level 2 until its length becomes <length>.

**Notes:** The pad string is repeated as many times as they are required to fit into <length>, eventually 0. If the input string is greater than <length>, it is trimmed to the right to fit in <length>.

**TOOLDIR Name:** ->torf

**Type:** Code

**Xlib #** 252

**Input:** TRUE or FALSE

**Output:**  
1 or 0

**Purpose:** Converts a system boolean in a numeric boolean.

**Notes:** Externals other than 3AC0 and 3A81 will cause an error "Bad Argument Value".

<b>TOOLDIR Name:</b>	alfaLext\$
<b>Type:</b>	String
<b>Xlib #</b>	251
<b>Input:</b>	none
<b>Output:</b>	String
<b>Purpose:</b>	Contains all lowercase accented characters (ECMA 94).

<b>TOOLDIR Name:</b>	alfaUext\$
<b>Type:</b>	System RPL
<b>Xlib #</b>	250
<b>Input:</b>	none
<b>Output:</b>	
	String
<b>Purpose:</b>	Contains all uppercase accented characters (ECMA 94).



**TOOLDIR Name:** splito

**Type:** System RPL

**Xlib #** 249

**Input:**

$\{ \text{Obj}_1 \dots \text{Obj}_{p-1} \}$   $\text{Obj}$   $\text{Obj}_{p+1} \dots \text{Obj}_n$   $\text{Obj}$

**Output:**

$\{ \text{Obj}_1 \dots \text{Obj}_{p-1} \}$   $\text{Obj}$   $\{ \text{Obj}_{p+1} \dots \text{Obj}_n \}$

**Purpose:** Splits the list in three chunks.

**Notes:** The command works also for Program Objects and for Algebraics.

**Uses:** splitl

**TOOLDIR Name:** split\$

**Type:** System RPL

**Xlib #** 248

**Input:**

String<sub>1</sub> String<sub>2</sub>

**Output:**

String<sub>1</sub>(1,p-1) String<sub>2</sub> String<sub>1</sub>(p+1) TRUE

or

String<sub>1</sub> String<sub>2</sub> FALSE

**Purpose:** Splits the string in three chunks with the middle chunk consisting of String<sub>2</sub>.

**Notes:** If String<sub>2</sub> is not contained in String<sub>1</sub>, strings are left unchanged and the boolean FALSE is returned. Otherwise split is performed and the boolean TRUE signals the success. Null strings are returned when necessary, but input strings cannot be null otherwise an error is issued.

**Uses:** splith

**TOOLDIR Name:** polygon

**Type:** System RPL

**Xlib #** 247

**Input:**

(x,y) radius stangle<sup>3</sup> endangle<sup>3</sup> step<sup>3</sup> <thick> <fill>

**Output:** Updates PICT

**Purpose:** Draws a polygon in PICT and optionally fills its area.

**Notes:** stangle<sup>3</sup> must be less than endangle<sup>3</sup>. step<sup>3</sup> can be 0, in this case the routine computes the angle required to generate an arc. <thick> must fall in the range defined on page 35. <fill> represents the fill pattern number type. When needed it is automatically reduced to modulo n<sub>fill</sub> depending on the size of list **patterns**. A value of <0> means no fill. Note that area fills is recommended only when the contour of the area is closed. The routine does not try to check this condition anyway. Moreover it is required that fill seed point falls within the physical boundaries of PICT. Seed point has the following cylindrical coordinates:  
SEED(radius/2, (stangle<sup>3</sup>+endangle<sup>3</sup>)/2).

Fill patterns are listed starting from page 41. For the generation of circles the value of 0 as step<sup>3</sup> is the safest choice. Nevertheless if your plot maintains constant size, i.e. PICT dims does not change from time to time, you may find a well tempered stepping angle that lets you plot the circle faster. Note that, increasing the thickness of lines, plotting time grows little. In order to draw squares, pentagons or some other polygon, obviously it is enough to calculate step<sup>3</sup> as 360/sides. If you need some rotation of the figure to be drawn, you must add (or subtract) the rotation angle to stangle<sup>3</sup> and to endangle<sup>3</sup>. Please note that radius represents the radius of the circle surrounding the polygon and not the apothem of the polygon. If you want to draw a square with sides of length 2, you must specify a radius of length 1.414213...

Outstanding lines or segments are automatically trimmed. To calculate the position of the polygon, PPAR scale values are used. If you defined an anisotropic scale in PPAR, figures will result stretched. This functionality (dispite of built-in ARC) lets you draw ellipses when the scale is anisotropic. For example, if you need to reproduce your plot on the IR printer that has a pixel ratio of 1.25, you can compensate the vertical stretch

issuing 1.25 \*H and drawing the plot with these new scale factors.

If PPAR or PICT are not yet defined, the routine calls PPARdef by itself.

**Uses:**

PPARdef, line, linetypes, fill, patterns, Todd.

**TOOLDIR Name:** PPARdef

**Type:** System RPL

**Xlib #** 245

**Input:** none

**Output:**

List

**Purpose:** If the variable exists, retrieves current PPAR value otherwise creates default PPAR in current directory and returns its value.

**Notes:** PPAR default values have been corrected to match current PICT size. Original Rom code did not support consistent default values, but initialized scale always to (-6.5,-3.2) (6.5,3.1), no matter about PICT dimensions. This routine lets you to initialize safely graphics in one shot with isometric scale if PPAR has not been yet defined and automatic PICT creation if needed. Typical usage of this routine consists in putting it at the top of user-defined graphics commands to assure the presence of PICT and PPAR as well.

**TOOLDIR Name:** mcentr\$

**Type:** System RPL

**Xlib #** 245

**Input:**

String <len>

**Output:**

String

**Purpose:** Centers strings separated by Linefeeds within blank strings of length <len>.

**Notes:** <len> must be greater than 0. Leading and trailing blanks in the source string are automatically trimmed.

**Uses:** center\$, lines->, ->lines, mtop.

**TOOLDIR Name:** center\$

**Type:** System RPL

**Xlib #** 244

**Input:**

String <len>

**Output:**

String

**Purpose:** Centers a string within a blank string of length <len>.

**Notes:** <len> must be greater than 0. Leading and trailing blanks in the source string are automatically trimmed.

**Uses:** splith, trim\$.

**TOOLDIR Name:** expbuf

**Type:** System RPL

**Xlib #** 243

**Input:**

#buffer <x> <y> <flag>

**Output:**

#buffer <x> <y> <flag>

**Purpose:** Expands buffer by 11 nibbles and stores the point.

**Notes:** If not enough memory, issues an error.

**Uses:** addp



**TOOLDIR Name:** scanp

**Type:** System RPL

**Xlib #** 242

**Input:**

#buffer Grob Grob(8x8) <xmax> <xmin> <y> T/F <flag>

**Output:**

#buffer Grob Grob(8x8) <xmax> <xmin> <y> T/F <flag>

**Purpose:** Scans new points.

**Notes:** Used by fill.

**Uses:** addp, expbuf, scan, scanp.

**TOOLDIR Name:** red

**Type:** System RPL

**Xlib #** 241

**Input:**

List

**Output:**

List

**Purpose:** Deletes redundant objects from the list.

**Notes:** checks empty lists.

**TOOLDIR Name:** diff

**Type:** System RPL

**Xlib #** 240

**Input:**

List<sub>1</sub> List<sub>2</sub>

**Output:**

List<sub>1-1&2</sub>

**Purpose:** Takes the set difference of the two given sets.

**Notes:** Checks empty lists and eliminates redundant data.

**TOOLDIR Name:** inter

**Type:** System RPL

**Xlib #** 239

**Input:**

List<sub>1</sub> List<sub>2</sub>

**Output:**

List<sub>1&2</sub>

**Purpose:** Takes the set intersection of the two given sets.

**Notes:** checks empty lists and eliminates redundant data.

**TOOLDIR Name:** union

**Type:** System RPL

**Xlib #** 238

**Input:**

List<sub>1</sub> List<sub>2</sub>

**Output:**

List<sub>1!2</sub>

**Purpose:** Takes the set union of the two given sets.

**Notes:** checks empty lists and eliminates redundant data.

**Uses:** red.

**TOOLDIR Name:** linetypes  
**Type:** List  
**Xlib #** 237  
**Input:** none  
**Output:** none  
**Purpose:** Contains 24 line patterns (linetypes).  
**Notes:** All grobs are 192x1.

Pattern #1

```
*****  
*****  
*****
```

Pattern #2

```
* * * * *  
* * * * *  
* * * * *
```

Pattern #3

```
* * * * *  
* * * * *  
* * * * *
```

Pattern #4

```
* * * * *  
* * * * *  
* * * * *
```

Pattern #5

```
** ** ** **  
** ** ** **  
** ** ** **
```

Pattern #6

```
*** *** ***  
*** *** ***  
*** *** ***
```

Pattern #7

```
*****  
*****  
*****
```

Pattern #8

```
*   ***   *   ***   *   ***   *   ***   *   ***   *   ***   *   ***
*   ***   *   ***   *   ***   *   ***   *   ***   *   ***   *   ***
*   ***   *   ***   *   ***   *   ***   *   ***   *   ***   *   ***
```

Pattern #9

```
*   * * * * *   *   * * * * *   *   * * * * *   *   * * * * *   *   * * * * *   *   * * * * *   *   * * * * *
*   * * * * *   *   * * * * *   *   * * * * *   *   * * * * *   *   * * * * *   *   * * * * *   *   * * * * *
*   * * * * *   *   * * * * *   *   * * * * *   *   * * * * *   *   * * * * *   *   * * * * *   *   * * * * *
```

Pattern #10

```
*       *       *       *       *       *       *       *       *       *       *       *       *       *       *
***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***
*       *       *       *       *       *       *       *       *       *       *       *       *       *       *
```

Pattern #11

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*       *       *       *       *       *       *       *       *       *       *       *       *       *       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Pattern #12

```
*****
**   **   **   **   **   **   **   **   **   **   **   **   **   **   **   **   **
*****
```

Pattern #13

```
*****
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*****
```

Pattern #14

```
*   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***
***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***
**   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   *
```

Pattern #15

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Pattern #16

```
*   * * * * *   * * * * *   * * * * *   * * * * *   * * * * *   * * * * *   * * * * *   * * * * *
*       *       *       *       *       *       *       *       *       *       *       *       *       *
*****   *****   *****   *****   *****   *****   *****   *****
```

Pattern #17

```
*****

*****
```

Pattern #18

```
*****
  *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
 *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
```

Pattern #19

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Pattern #20

```
  *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
*   *   *   *   *   *   *   *   *   *   *   *   *   *   *
  *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
```

Pattern #21

```
  *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
  *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
```

Pattern #22

```
  *           *           *           *           *           *           *
 ***         ***         ***         ***         ***         ***         ***
  *           *           *           *           *           *           *
```

Pattern #23

```
  * * * * *   * * * * *   * * * * *   * * * * *   * * * * *   * * * * *   * * * * *
*   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
  * * * * *   * * * * *   * * * * *   * * * * *   * * * * *   * * * * *
```

Pattern #24

```
*****
```



**TOOLDIR Name:** box

**Type:** System RPL

**Xlib #** 236

**Input:**

GROB GROB<sub>pattern</sub> <x1> <y1> <x2> <y2> <thick> <mask>

**Output:**

GROB GROB<sub>pattern</sub> <x1> <y1> <x2> <y2> <thick> <mask>

**Purpose:** Draws a box using specified line pattern and thickness.

**Notes:** Pairs of coordinates can be given in any order. Parts of the box lying outside physical bounds of GROB are automatically clipped.  
Thickness can't exceed 3.  
Line patterns are automatically mirrored (only if thick>1).  
Allowed values for mask can be found under word **line**.

**Uses:** line

**TOOLDIR Name:** gop  
**Type:** CODE  
**Xlib #** 235

**Input:**

GROB1 (WxH) GROB2 (WxH) <oper>

**Output:**

GROB1 (WxH)

**Purpose:** Applies operator between GROBs and stores the result in GROB1.

**Notes:** Operator is so defined:

<0>	:	null	keeps GROB1
<1>	:	replace	stores GROB2 onto GROB1
<2>	:	exor	
<4>	:	or	
<8>	:	and	
<16>	:	reserved	
<32>	:	reserved	
<64>	:	reserved	
<128>	:	reserved	

**WARNING: GROB1 cannot reside in ROM.**

**TOOLDIR Name:** line

**Type:** CODE

**Xlib #** 234

**Input:**

GROB GROB<sub>pattern</sub> <x1> <y1> <x2> <y2> <thick> <mask>

**Output:**

GROB GROB<sub>pattern</sub>

**Purpose:** Draws a line of specified thickness using the pattern supplied.

**Notes:** Points can be given in any order. Numbers are represented in two's complement, that is -1 is equivalent to <FFFFFF>. You can use negative coordinates for both x and y values. Theoretical range is -524288 to 524287 for both axes. However you must consider that grobs are not virtual (!). Reserved values should not be used.

If <thick> is greater than 8 the line is mirrored.

If you plan to use thick=<1>, you can supply a GROB<sub>pattern</sub> of 64x1. By analogy, when drawing with thick=2, it's enough a GROB<sub>pattern</sub> of size 128x1.

See predefined line patterns on page 30 as example. Pattern alignment is relative to the starting point and not to the origin.

## MASK and THICKNESS Reference Table

### mask      applies

<0> :      null (draws nothing).  
 <1> :      replace  
 <2> :      exor  
 <4> :      or  
 <8> :      and  
 <16> :     reserved  
 <32> :     reserved  
 <64> :     reserved  
 <128>:     reserved

### thick      draws

<p>           &lt;0&gt;            nothing            &lt;1&gt;            single line            &lt;2&gt;            double line         </p>	<p>           main line uses first 8 bytes of GROB pattern, while the second line is drawn incrementing X or Y and uses 8 remaining bytes.         </p>
<p>           &lt;3&gt;            triple line         </p>	<p>           middle line uses first 8 bytes of GROB pattern, inner line uses next 8 bytes, outer line uses remaining 8 bytes.         </p>
<p>           &lt;4&gt;            reserved            &lt;5&gt;            reserved            &lt;6&gt;            reserved            &lt;7&gt;            reserved            &lt;8&gt;            nothing            &lt;9&gt;            single line            &lt;A&gt;            double line         </p>	<p>           main line uses first 8 bytes of GROB pattern, while the second line is drawn decrementing X or Y and uses 8 remaining bytes (mirror).         </p>
<p>           &lt;B&gt;            triple line         </p>	<p>           middle line uses first 8 bytes of GROB pattern, outer line uses next 8 bytes, inner line uses remaining 8 bytes (mirror).         </p>

**TOOLDIR Name:** rect

**Type:** System RPL

**Xlib #** 233

**Input:**

GROB GROB<sub>line</sub> GROB<sub>fill</sub> <x1> <y1> <x2> <y2> <thick> <mask>

**Output:**

GROB

**Purpose:** Draws a box filled with the specified pattern.

**Notes:** Pairs of coordinates can be given in any order. Parts of the box lying outside physical bounds of GROB are automatically clipped. <mask> works in the same way for both lines and fill. Allowed values for mask and can be found under word **line**. <thick> cannot exceed <3>.

**Uses:** box, gop, vfill

**TOOLDIR Name:** ord

**Type:** CODE

**Xlib #** 232

**Input:**

<h1> <h2>

**Output:**

<max> <min>

**Purpose:** Orders two system binaries.

**TOOLDIR Name:** popp

**Type:** CODE

**Xlib #** 231

**Input:**

#stack

**Output:**

#stack <x> <y> <flags> TRUE

or

#stack FALSE

**Purpose:** Pops a pair of binary integers and 4 bit user info from a private stack.

**Notes:** The opposite function is addp. Lowest 4 bits of <flags> may contain some user information.

**TOOLDIR Name:** addp

**Type:** CODE

**Xlib #** 230

**Input:**

#stack      <x>              <y>              <flags>

**Output:**

#stack      <x>              <y>              <flags>      T/F

**Purpose:** Pushes two binary integers on a private stack.

**Notes:** This routine is currently used to push points (pairs of binary integers) on a stack for subsequent evaluation during fill. Nevertheless it can be used as a convenient LIFO buffer for some other purpose. If the stack is full, numbers are not stored and a FALSE flag is returned allowing stack enlargement or error generation. <flags> are stored together with <x> and <y> and represent some user condition. Only the least significant nibble is stored, that is you have only 4 bits available in <flags>. The opposite function is popp.



**TOOLDIR Name:** patterns

**Type:** List

**Xlib #** 229

**Input:** none

**Output:** none

**Purpose:** Contains 45 different fill patterns (textures).

**Notes:** All pattern grobs are 8x8.

Pattern #1

```

*****
*****
*****
*****
*****
*****
*****
*****

```

Pattern #2

```

*****
*****
*****
*****
*****
*****
*****
*****

```

Pattern #3

```

** ** ** **
** ** ** **
** ** ** **
** ** ** **
** ** ** **
** ** ** **
** ** ** **
** ** ** **

```

Pattern #4

```

** **
** **
** **
** **
** **

```

Pattern #5

```

*****
*****
*****
*****
*****
*****
*****
*****

```

Pattern #6

```

*****
** ** ** **
*****
** ** ** **
*****
** ** ** **
*****
** ** ** **

```

Pattern #7

```

** ** ** **
** ** ** **
** ** ** **
** ** ** **

```

Pattern #8

```

** **
** **
** **
** **
** **
** **

```

Pattern #9

```

*****
** ** ** **
*****
*****
*****
** ** ** **
*****
*****

```

Pattern #10

```

    **      **
      **      **
**      **
    **      **
      **      **
    **      **
**      **

```

Pattern #11

```

*****  *****
*****  *****  **
**  *****  *****
    *****  *****
*****  *****
*****  *****  **
**  *****  *****
    *****  *****

```

Pattern #12

```

*****
*****
*****
*****
*****

```

Pattern #13

```

**  **  **  **
**  **  **  **
**  **  **  **
**  **  **  **
**  **  **  **
**  **  **  **
**  **  **  **
**  **  **  **

```

Pattern #14

```

    **      **
      **      **
**  **  **  **
    **      **
      **      **
    **      **
**  **  **  **
**  **  **  **

```

Pattern #15

```

*****  *****  **
**  *****  *****
    **  **  **  **
*****  *****
*****  *****  **
**  *****  *****
    **  **  **  **
*****  *****

```

Pattern #16

```

    **      **
      **      **
**      **
    **  **  **  **
      **      **
    **      **
**      **
    **  **  **  **

```

Pattern #17

```

*****  *****  **
**  *****  *****
    *****  *****
**  **  **  **
*****  *****  **
*****  *****
*****  *****
**  **  **  **

```

Pattern #18

```

*****
*****
**      **
*****
    **      **
*****
**      **
*****

```

Pattern #19

```

**  *****  **
*****  *****
**  *****  **
*****  *****

```

Pattern #20

```

    **
    **
    **
*****
    **
    **
    **
*****

```

Pattern #21

```

*****  *****
*****  *****
*****  *****
*****  *****
*****  *****
*****  *****

```

Pattern #22

```

    **      **
      **      **
*****
    **      **
    **      **
    **      **
*****
    **      **

```

Pattern #23

```

*****  *****
*****  *****
*****  *****
*****  *****
*****  *****
*****  *****

```

Pattern #24

```

**  **  **  **
    **      **
**  **  **  **
    **      **
**  **  **  **
    **      **
**  **  **  **
    **      **

```

Pattern #25

```

  **  **  **  **
**  * * * * *  * * * *
  **  **  **  **
* * * * *  * * * *
  **  **  **  **
**  * * * * *  * * * *
  **  **  **  **
* * * * *  * * * *

```

Pattern #26

```

**  **
  **
**  **
      **      **
          **  **
              **
          **  **
      **      **

```

Pattern #27

```

  **  * * * * * * * * *
**  * * * * * * * * *
  **  * * * * * * * * *
* * * * *  * * * *
* * * * *  **  **
* * * * * * * * *  * * * *
* * * * *  **  **
* * * * *  * * * *

```

Pattern #28

```

**          **
      **      **
**          **
      **      **

```

Pattern #29

```

  * * * * *  * * * * *
* * * * *  * * * * *
* * * * * * * * * *
* * * * * * * * * *
  * * * * *  * * * * *
* * * * *  * * * * *
* * * * * * * * * *
* * * * * * * * * *

```

Pattern #30

```

* * * * *
**  * * * *      **
* * * * *
      **  **  **  **
          * * * * *
          * * * *  * * * *
          * * * * *
      **  **  **  **

```

Pattern #31

```

  * * * * * * * * *
  **  * * * * *
      * * * * * * * * *
**  **  **  **
* * * * * * * * *  **
* * * * *      **
* * * * * * * * *  **
**  **  **  **

```

Pattern #32

```

  **  **
  **          **
**  **      **  **
      * * * *
      * * * *
**  **      **  **
  **          **
  **      **

```

Pattern #33

```

* * * *  * * * *  * * * *
**  * * * * * * * *  **
      * * * *  **
* * * * *      * * * * *
* * * * *      * * * * *
      * * * * *  **
**  * * * * * * * *  **
* * * *  * * * *  * * * *

```

Pattern #34

```

      * * * *
  * * * *  **  **
**  **      * * * *
* * * *  * * * *
      * * * *  * * * *
  * * * *      **  **
  **  **  * * * *
      * * * *

```

Pattern #35

```

* * * * * * * * *  * * * *
**  **      **  **  **
  **  * * * *      **
      **      * * * * *
* * * * *      **
**  **      * * * *  **
**  **  **  **      **
* * * *      * * * * *

```

Pattern #36

```

      * * * *
      * * * *  **  **
**  **  * * * * *
* * * * *
          * * * * *
      * * * * *  **  **
      **  **  * * * *
      * * * *

```

Pattern #37

```

* * * * * * * * *  * * * *
**  **      **  **
  **  **      **
      * * * * * * * * *
* * * * * * * * *
**  **      **  **
**  **  **  **      **
* * * *      * * * * *

```

Pattern #38

```

      **
      **
      **
      **  **
* * * *  **  * * * * *
      **  **
      **
      **

```

Pattern #39

```

* * * * *  * * * * *
* * * * *  * * * * *
* * * * *  * * * * *
* * * *  **  * * * * *
      **  **
* * * *  **  * * * * *
* * * * *  * * * * *
* * * * *  * * * * *

```

Pattern #40

```

**
  **              **
    *****
  *****
    *****
      **
    *****
  *****
    *****
**              **
```

Pattern #43

```

*****
*****
*****

**      *****      **
**      *****      **
**      *****      **
```

Pattern #41

```

*****
** *****
*****      **      **
*****      **      **
*****      **      **
*****      **      **
*****      **      **
** *****
```

Pattern #44

```

  **
    **      **
      **
    **
  **      **
    **
```

Pattern #42

```

      **      **
      **      **
      **      **
*****
      **      **
      **      **
      **      **
*****
```

Pattern #45

```

*****
*****
*****      **
*****      **
*****      **
*****      **
**      *****
*****      **
*****
```

**TOOLDIR Name:** fill

**Type:** System RPL

**Xlib #** 228

**Input:**

Grob      Grob      pattern      <x>      <y>

**Output:**

Grob

**Purpose:** Fills a region with the specified pattern.

**Notes:** The pattern Grob must be at least 8x8 or equivalent (i.e. 64x1). You can fill either black or white regions. There is no restriction on the shape of the area being filled. The algorithm adjusts memory requirements automatically. It is possible to fill black or white regions. In low memory conditions, the process could become slow due to garbage collection or even abort the filling when insufficient memory is detected.

**Uses:** addp, popp, rpt, scan.

**TOOLDIR Name:** vfill

**Type:** System RPL

**Xlib #** 227

**Input:**

Grob Grob(8x8) <x> <y> <lasty>

**Output:**

Grob

**Purpose:** Fills a region with the specified pattern, provided that <y> represents the lowest coordinate of the region to be filled (upper corners of Display) and <lasty> the bottom row.

**Notes:** The pattern Grob must be exactly 8x8. You can fill either black or white regions. <x> may range freely between left and right bound however it must fall within the region. This routine can fill regions of irregular shape such that each middle point between left and right bounds can be a valid starting point for next fill. In other terms a point below the middle point of the previous line must have the same color.

**Uses:** scan.

**TOOLDIR Name:** rtrim\$

**Type:** System RPL

**Xlib #** 226

**Input:**

Str

**Output:**

Str

**Purpose:** Trims blanks at right side of string.

**Notes:** none.

**Uses:** ltrim\$, revs.

**TOOLDIR Name:** ltrim\$

**Type:** System RPL

**Xlib #** 225

**Input:**

Str

**Output:**

Str

**Purpose:** Trims blanks at left side of string.

**Notes:** none.

**Uses:** span\$.



**TOOLDIR Name:** lwc\$

**Type:** System RPL

**Xlib #** 224

**Input:**

Str <start>

**Output:**

Str

**Purpose:** Converts all uppercase letters to lowercase starting from <start>.

**Notes:** <start> must be greater than 0.

**Uses:** splith, member\$, span\$.

**TOOLDIR Name:** upc\$

**Type:** System RPL

**Xlib #** 223

**Input:**

Str <start>

**Output:**

Str

**Purpose:** Converts all lowercase letters to uppercase starting from <start>.

**Notes:** <start> must be greater than 0.

**Uses:** member\$, span\$, splith.

**TOOLDIR Name:** norm\$

**Type:** System RPL

**Xlib #** 222

**Input:**

Str

**Output:**

Str

**Purpose:** Creates a new string from the original, deleting exceeding blanks between words and trimming leading and trailing blanks.

**Notes:** none.

**Uses:** trim\$, splith.

**TOOLDIR Name:** trim\$

**Type:** System RPL

**Xlib #** 221

**Input:**

Str

**Output:**

Str

**Purpose:** Trims leading and trailing blanks from the string.

**Notes:** none.

**Uses:** ltrim\$, rtrim\$.

**TOOLDIR Name:** ->msg\$

**Type:** System RPL

**Xlib #** 220

**Input:**

<LID> <msg#>

**Output:**

Str

**Purpose:** Retrieves the corresponding message from specified Library message table.

**Notes:** If library does not exist or msg# out of range, returns "". It must be 0 < msg# < 256d.

**TOOLDIR Name:** revb

**Type:** Code

**Xlib #** 219

**Input:**

Binary

**Output:**

Binary

**Purpose:** Reverses the order of the nibbles in the hex string.

**Notes:** none.

**TOOLDIR Name:** rev\$

**Type:** Code

**Xlib #** 218

**Input:**

Str

**Output:**

Str

**Purpose:** Reverses the order of the characters in the string.

**Notes:** none.

**TOOLDIR Name:** Xlib->

**Type:** Code

**Xlib #** 217

**Input:**

Xlib

**Output:**

<LID> <num>

**Purpose:** Splits Xlib name information.

**Notes:** The opposite function (->xlib) is located at #07E50 in the internal ROM.



**TOOLDIR Name:** ->prolog

**Type:** Code

**Xlib #** 216

**Input:**

Obj

**Output:**

<Prolog>

**Purpose:** Retrieves object's prolog.

**Notes:** none

**TOOLDIR Name:** span

**Type:** Code

**Xlib #** 215

**Input:**

String\$ Set\$ <s>

**Output:**

<p>

**Purpose:** Seeks the first character not comprised in the set of characters given, starting from position s.

**Notes:** p is an absolute position (not relative to s).

**TOOLDIR Name:** member

**Type:** System RPL

**Xlib #** 214

**Input:**

String\$ Set\$ <s>

**Output:**

<p>

**Purpose:** Seeks the first character member of the set of characters given, starting from position <s>.

**Notes:** p is an absolute position (not relative to s).

**TOOLDIR Name:** scan

**Type:** Code

**Xlib #** 213

**Input:**

Grob Grob(8x8) <x> <y> T/F

**Output:**

Grob Grob(8x8) <Xright> <Xleft>

**Purpose:** Searches right and left bounds, that is returning the position of last pixels with the same color respect to the origin and if flag is TRUE fills the line between bounds with the specified pattern.

**Notes:** Grob must be not larger than 2048 x 4095 pixels (theoretically). The pattern Grob must be exactly 8x8. You can fill either black or white regions (lines). Pattern is automatically aligned respect to the origin. This means that consecutive lines are filled with different pattern alignment. This is necessary to preserve the shape of the pattern independently of the relative position of the area being filled.

**TOOLDIR Name:** npos

**Type:** System RPL

**Xlib #** 212

**Input:**

Comp { } <maxd> <1> Prg Obj

**Output:**

Comp List(lists of sysbin)

**Purpose:** Reports all the positions in the compound object where each element combined with Obj, by mean of Prg, produces a True condition.

**Notes:** The format of resulting positions is good for nget and nput. The <maxd> parameter specifies the maximum search depth. By putting <0> as <maxd>, search is limited only to RAM level compound objects. This feature prevents an endless loop when compound objects are recursive. If you want to unthread ROM objects, you must specify a maximum search depth.  
Specifying 'Same' as Prg you get all the occurrences of Obj within Comp. Specifying a program like the following, you get all the occurrences of Global Names:

```
Prog
  Nip
  T_If_Global
End
```

In this case Obj is a dummy object. This technique can be useful for creating a simple (and effective) cross reference program (see in the Application Disc of the SmartROM).

**Uses:** ncount, mrev, TifC.

**TOOLDIR Name:** ncount

**Type:** System RPL

**Xlib #** 211

**Input:**

<n>

**Output:**

meta<sub>h</sub>(sysbin)

**Purpose:** Retrieves the current values of the last n open loops.

**Notes:** A check is made on how many loops are currently open. (Bad Arg Value issued). The loops are arranged in the following order:  
The inner loop is the first object in the meta (highest stack level) while the outer one comes last (lowest stack level). This command avoids tedious stack manipulation when you need many counters at the same time (typically in matrix algorithms).

**Uses:** getaddr.

**TOOLDIR Name:** getaddr

**Type:** Code

**Xlib #** 210

**Input:**

<addr>

**Output:**

<@addr>

**Purpose:** Pushes on stack the contents of the addr given.

**Notes:** none.

**TOOLDIR Name:** TifC

**Type:** Code

**Xlib #** 209

**Input:**

Obj

**Output:**

T/F

**Purpose:** Returns TRUE if Obj is a compound object, FALSE otherwise.

**Notes:** TRUE if type is either Program or Algebraic or List.



**TOOLDIR Name:** putobj

**Type:** System RPL

**Xlib #** 208

**Input:**

Comp      <pos>      Obj

**Output:**

Comp

**Purpose:** Puts an object into a compound object.

**Notes:** Error (Bad Arg Val) if an invalid path is detected.

**TOOLDIR Name:** nput

**Type:** System RPL

**Xlib #** 207

**Input:**

Comp {<dim<sub>1</sub>> <dim<sub>2</sub>> ..... <dim<sub>n</sub>>} Obj

**Output:**

Comp

**Purpose:** Puts an object into a nested compound object.

**Notes:** If the sequence is meaningless issues Bad Arg Value. No restrictions affect the shape of the object.

**Uses:** RD, rptcmd, mrev, TifC, putobj.

**TOOLDIR Name:** nget

**Type:** System RPL

**Xlib #** 206

**Input:**

Comp {<dim<sub>1</sub>> <dim<sub>2</sub>> .... <dim<sub>n</sub>>}

**Output:**

Obj<sub>dim1,dim2,...,dimn</sub>

**Purpose:** Gets a nested element from a list.

**Notes:** If the sequence is meaningless issues Bad Arg Value. No restrictions affect the shape of the list.

**Uses:** rptcmd, TifC, mrev.

**TOOLDIR Name:** apply

**Type:** System RPL

**Xlib #** 205

**Input:**

ListMat (m,n)                      Obj                      Prg

**Output:**

ListMat (m,n)

**Purpose:** Combines each element of the matrix with the given object by means of the binary operator prg and puts the result in a new matrix.

**Notes:** prg must be a binary procedure returning a single result. If You use '+' as Prg in conjunction with numeric or symbolic values, you get ADDCON. You could also think to 'apply' Eval (or ->Num) to each element of the matrix (remember to drop Obj !), thus obtaining a recalc feature.

**Uses:** dims, mtop.

**TOOLDIR Name:** ->ext

**Type:** Code

**Xlib #** 204

**Input:**

<addr>

**Output:**

Obj

**Purpose:** Returns the object pointed to by <addr> (if any).

**Notes:** <addr> must be less than 70000h (else Port Not Available error is issued).

**TOOLDIR Name:** revsys

**Type:** System RPL

**Xlib #** 203

**Input:**

<n> <bits>

**Output:**

<m>

**Purpose:** Reverses the bit order in <n>.

**Notes:** <bits> must be less than <14h> (20d).

**TOOLDIR Name:** ma2

**Type:** System RPL

**Xlib #** 202

**Input:**

meta<sub>h1</sub>      meta<sub>h2</sub>

**Output:**

meta<sub>h1+h2</sub>

**Purpose:** Adds two meta-objects.

**Notes:** No check is made to ensure Stack integrity.

**TOOLDIR Name:** metax

**Type:** System RPL

**Xlib #** 201

**Input:**

meta<sub>h</sub>      <i>      <j>

**Output:**

meta<sub>h</sub>

**Purpose:** Swaps element i and element j of the meta.

**Notes:** Does not check ranges.

**Uses:** xlev.



**TOOLDIR Name:** getcol

**Type:** System RPL

**Xlib #** 200

**Input:**

ListMat(m,n) <c>

**Output:**

{ obj<sub>1c</sub> obj<sub>2c</sub> ... obj<sub>nc</sub> }

**Purpose:** Retrieves a column from a matrix.

**Notes:** 0 means Replace Cursor while 1 means Insert Cursor.  
curpos, row and col may be 0 (Last position).

**Uses:** mtop.

**TOOLDIR Name:** VISIT

**Type:** System RPL

**Xlib #** 199

**Input:**

curpos 0/1 Global

or

{ row col} 0/1 Global

or

"search\_key" occur Global

**Output:**

**Purpose:** Calls the editor with some parameters.

**Notes:** 0 means Replace Cursor while 1 means Insert Cursor.  
curpos, row and col may be 0 (Last position).

**Uses:** CKL2R, search, visit.

**TOOLDIR Name:** visit

**Type:** System RPL

**Xlib #** 198

**Input:**

<curpos> <curst> Global

or

{ <row> <col>} <curst> Global

**Output:**

**Purpose:** Calls the editor with some parameters.

**Notes:** <curst> may be <1> (Insert) or <2> (Replace). <curpos>, <row> and <col> may be <0> (Last position).

**TOOLDIR Name:** !ckr

**Type:** System RPL

**Xlib #** 197

**Input:**

$\{ R_1 R_2 \dots R_n \}$

**Output:**

$\{ R_1 R_2 \dots R_n \}$

**Purpose:** Checks if the list contains only real numbers. If not, errors out.

**Notes:** none

<b>TOOLDIR Name:</b>	!ckrol
<b>Type:</b>	System RPL
<b>Xlib #</b>	196
<b>Input:</b>	
	List
<b>Output:</b>	
	List
<b>Purpose:</b>	Checks if the list contains real numbers or lists of reals. If not, errors out.
<b>Notes:</b>	none
<b>Uses:</b>	!ckr.

<b>TOOLDIR Name:</b>	#key->\$
<b>Type:</b>	System RPL
<b>Xlib #</b>	195
<b>Input:</b>	Keyboard Input
<b>Output:</b>	A character in Edit Mode.
<b>Purpose:</b>	Outputs a character given its ASCII code, interactively.
<b>Notes:</b>	Checks if ASCII code is meaningful (1-255).
<b>Uses:</b>	#k.

**TOOLDIR Name:** #k  
**Type:** System RPL  
**Xlib #** 194

**Input:**

{ <k<sub>1</sub>> <k<sub>2</sub>> ... <k<sub>n</sub>> }

**Output:**

<listpos>

**Purpose:** Waits for one of the keys specified in the list and returns its corresponding position. <0> means not found.

**Notes:** none

<b>TOOLDIR Name:</b>	\$&LMENU
<b>Type:</b>	List
<b>Xlib #</b>	193
<b>Input:</b>	none
<b>Output:</b>	none
<b>Purpose:</b>	Contains String & List commands menu list.
<b>Notes:</b>	none



**TOOLDIR Name:** \$CONFIG

**Type:** Program

**Xlib #** 192

**Input:** none

**Output:** none

**Purpose:** Attaches Library <335> to the HOME directory.

**Notes:** none

**TOOLDIR Name:** BIND

**Type:** System RPL

**Xlib #** 191

**Input:**

List { Obj<sub>o</sub> Obj<sub>n</sub> }

**Output:**

List

**Purpose:** Substitutes Obj<sub>o</sub> with Obj<sub>n</sub> in List.

**Notes:**

**Uses:** replace.

**TOOLDIR Name:** CHECKL

**Type:** System RPL

**Xlib #** 190

**Input:**

{...} Type\_number

**Output:**

{...}

Error if type mismatch.

**Purpose:** Check List contents. All the objects contained in the list must be of same type.

**Notes:** Type\_number is a real number in the range 0-27 (same as TYPE command values).

**TOOLDIR Name:** CKOBJ

**Type:** Program

**Xlib #** 189

**Input:**

meta

**Output:**

meta<sub>h</sub>

**Purpose:** Checks if all objects of the meta-object are of the same type.

**Notes:** Reference type is assumed to be the type of the last Obj.  
If type mismatches issues Bad Arg Value error.

**TOOLDIR Name:** CKL2R

**Type:** System RPL

**Xlib #** 188

**Input:**

{ Real1 Real2 }

**Output:**

2 : Real<sub>1</sub>  
1 : Real<sub>2</sub>

**Purpose:** Checks the input list for 2 Real numbers. An error is issued if type mismatches or list size not equal to 2.

**Notes:** none.

**Uses:** CHECKL.

<b>TOOLDIR Name:</b>	COPYRIGHT
<b>Type:</b>	Grob
<b>Xlib #</b>	187
<b>Input:</b>	none
<b>Output:</b>	none
<b>Purpose:</b>	Contains Copyright & ROM revision info.
<b>Notes:</b>	none

**TOOLDIR Name:** LFPOS

**Type:** System RPL

**Xlib #** 186

**Input:**

String <LF\_search\_start>

**Output:**

<pos>

**Purpose:** Find first non-embedded LF in the given string (if any).

**Notes:** none

**TOOLDIR Name:** MDS

**Type:** System RPL

**Xlib #** 185

**Input:**

$\text{Obj}_n \dots \text{Obj}_1 \dots \text{Obj}_f \dots \text{Obj}_d \dots \text{Obj}_1$     <l>    <f>    <d>

**Output:**

$\text{Obj}_n \dots \text{Obj}_{1+1} \text{Obj}_{f-1} \dots \text{Obj}_1 \dots \text{Obj}_f \text{Obj}_d \dots \text{Obj}_1$

**Purpose:** Moves downwards Stack objects.

**Notes:** Requires  $l > f > d$ .

**Uses:** rdown.



<b>TOOLDIR Name:</b>	METAMENU
<b>Type:</b>	List
<b>Xlib #</b>	184
<b>Input:</b>	none
<b>Output:</b>	none
<b>Purpose:</b>	Contains Meta-objects commands menu list.
<b>Notes:</b>	none

<b>TOOLDIR Name:</b>	MISCMENU
<b>Type:</b>	List
<b>Xlib #</b>	183
<b>Input:</b>	none
<b>Output:</b>	none
<b>Purpose:</b>	Contains Miscellaneous commands menu list.
<b>Notes:</b>	none

**TOOLDIR Name:** MUS

**Type:** System RPL

**Xlib #** 182

**Input:**

$\text{Obj}_n \dots \text{Obj}_d \dots \text{Obj}_1 \dots \text{Obj}_f \dots \text{Obj}_1$  <l> <f> <d>

**Output:**

$\text{Obj}_n \dots \text{Obj}_1 \dots \text{Obj}_f \text{Obj}_d \dots \text{Obj}_{1+1} \text{Obj}_{f-1} \dots \text{Obj}_1$

**Purpose:** Moves upwards Stack objects.

**Notes:** Requires  $d > 1 > f$ .

**Uses:** rdown, rup.

**TOOLDIR Name:** RD

**Type:** System RPL

**Xlib #** 181

**Input:**

$\text{obj}_1 \dots \text{obj}_n$        $\langle t \rangle$        $\langle n \rangle$

**Output:**

$\text{obj}_{n-t+1} \dots \text{obj}_1$        $\text{obj}_n \dots \text{obj}_{t-1}$

**Purpose:** Roll down  $\langle n \rangle$  objects  $\langle t \rangle$  times.

**Notes:** no check is made.

<b>TOOLDIR Name:</b>	RU
<b>Type:</b>	System RPL
<b>Xlib #</b>	180
<b>Input:</b>	
$obj_1 \dots obj_n$	$\langle t \rangle \qquad \langle n \rangle$
<b>Output:</b>	
$obj_{t-1} \dots obj_n$	$obj_1 \dots obj_t$
<b>Purpose:</b>	Roll up $\langle n \rangle$ objects $\langle t \rangle$ times.
<b>Notes:</b>	No check is made.

<b>TOOLDIR Name:</b>	STACKMENU
<b>Type:</b>	List
<b>Xlib #</b>	179
<b>Input:</b>	none
<b>Output:</b>	none
<b>Purpose:</b>	Contains Stack commands menu list.
<b>Notes:</b>	none

<b>TOOLDIR Name:</b>	SYMBMENU
<b>Type:</b>	List
<b>Xlib #</b>	178
<b>Input:</b>	none
<b>Output:</b>	none
<b>Purpose:</b>	Contains Symbolic Manipulation commands menu list.
<b>Notes:</b>	none

<b>TOOLDIR Name:</b>	Smartmenu
<b>Type:</b>	List
<b>Xlib #</b>	177
<b>Input:</b>	none
<b>Output:</b>	none
<b>Purpose:</b>	Contains menu of menu list & menu handling software.
<b>Notes:</b>	none



<b>TOOLDIR Name:</b>	TYPESMENU
<b>Type:</b>	List
<b>Xlib #</b>	176
<b>Input:</b>	none
<b>Output:</b>	none
<b>Purpose:</b>	Contains Type Conversion commands menu list.
<b>Notes:</b>	none

**TOOLDIR Name:** add

**Type:** System RPL

**Xlib #** 175

**Input:**

Symbmat<sub>1</sub> Symbmat<sub>2</sub>

**Output:**

Symbmat

**Purpose:** Add Symbmat<sub>1</sub> & Symbmat<sub>2</sub> Element by Element.

**Notes:** Both lists must have identical dimensions.

**Uses:** dot.

**TOOLDIR Name:** addcon

**Type:** System RPL

**Xlib #** 174

**Input:**

SymbMat Symbolic

**Output:**

SymbMat

**Purpose:** Add a constant to all elements.

**Notes:** none.

**Uses:** apply.

<b>TOOLDIR Name:</b>	addobj
<b>Type:</b>	System RPL
<b>Xlib #</b>	173
<b>Input:</b>	Same as +
Obj <sub>1</sub> Obj <sub>2</sub>	
<b>Output:</b>	Same as + except for Grobs and Binary Integers.
Obj	
<b>Purpose:</b>	Add two objects allowing Grob and Binary Integer linking.
<b>Notes:</b>	<p>The OR function for Grobs has been replaced by gaddr.  The add function for Binary Integers has been replaced by  append.  These changes makes RPT function useful for certain  purposes.</p>
<b>Uses:</b>	gaddr.

**TOOLDIR Name:** argerr

**Type:** System RPL

**Xlib #** 172

**Input:**

**Output:** Subscript Out of Range Error.

**Purpose:** Issues "Subscript Out of Range" error <33504h>.

**Notes:** none.

**TOOLDIR Name:** c2m

**Type:** System RPL

**Xlib #** 171

**Input:**

n : Obj  
...: ...  
m+2: Obj  
m+1: 'mark  
m : Obj  
...: Obj  
1 : Obj

**Output:**

n+1: Obj  
...: ...  
m+3: Obj  
m+2: 'mark  
m+1: Obj  
...: Obj  
2 : Obj  
1 : <m>

**Purpose:** Counts objects between TOS and private name 'mark.

**Notes:** Same as Depth if mark not found.

**Uses:** mark.

**TOOLDIR Name:** chl?

**Type:** System RPL

**Xlib #** 170

**Input:**

{ Obj<sub>1</sub> ... Obj<sub>n</sub> } {t<sub>1</sub> t<sub>n</sub>}

**Output:**

{ Obj<sub>1</sub> ... Obj<sub>n</sub> } { err<sub>1</sub> ... err<sub>m</sub> } TRUE

or

{ Obj<sub>1</sub> ... Obj<sub>n</sub> } FALSE

**Purpose:** Checks the list in level 2 for the given object types. The type specification can be either a real number or a list of reals.

**Notes:** If FALSE is returned, object types match the specifications, otherwise is returned TRUE and a list of positions where object types mismatch.

**Uses:** idx?, lop1, lvop.

**TOOLDIR Name:** chset?

**Type:** System RPL

**Xlib #** 169

**Input:**

{ Obj<sub>1</sub> Obj<sub>2</sub> ... Obj<sub>n</sub> } List\_of\_Types

**Output:**

{ err<sub>1</sub> ... err<sub>m</sub> } TRUE

or

FALSE

**Purpose:** Checks if objects contained in the list are among types specified in List\_of\_types.

**Notes:** Types specified in List\_of\_types are real numbers complying with TYPE function and are **not** positional, but global. Thus if you supply only a value in List\_of\_types, all objects in the first list must be of the same type. If FALSE is returned, object types match the specifications, otherwise is returned TRUE and a list of positions where object types mismatch.

Example:

Input: { 1 "" 2 } { 0 } 821 169 ->Xlib EVAL

Output: { 1 "" 2 } { 2 } FALSE

**Uses:** mtop, mtopn, rptcmd.



**TOOLDIR Name:** chst?

**Type:** System RPL

**Xlib #** 168

**Input:**

$\text{Obj}_1 \dots \text{Obj}_n$   $\{r_1 \dots r_n\}$

**Output:**

$\text{Obj}_1 \dots \text{Obj}_n$   $\{ \text{err}_1 \dots \text{err}_m \}$  TRUE

or

$\text{Obj}_1 \dots \text{Obj}_n$  FALSE

**Purpose:** Checks stack contents.

**Notes:** If FALSE is returned, object types match the specifications, otherwise is returned TRUE and a list of positions where object types mismatch.

**Uses:** chl?.

**TOOLDIR Name:** cmpl

**Type:** System RPL

**Xlib #** 167

**Input:**

SymbMat (n)                      <r>                      <c>

**Output:**

SymbMat (n-1)

**Purpose:** Returns the complement of the element (i,j).

**Notes:** To obtain the algebraic complement  $a_{sr}^*$  of the element  $a_{rs}$ , You must calculate the determinant of the resulting matrix and multiply it by  $(-1)^{(r+s)}$ .

**Uses:** delcol, delrow, square?.

**TOOLDIR Name:** conform?

**Type:** System RPL

**Xlib #** 166

**Input:**

Listmat<sub>1</sub> Listmat<sub>2</sub>

**Output:**

Listmat<sub>1</sub> Listmat<sub>2</sub> <m> <n> <n> <p> TRUE

or

ListMat<sub>1</sub> ListMat<sub>2</sub> FALSE

**Purpose:** Check matrix conformability.

**Notes:** Mat1(m,n), Mat2(n,p) -> TRUE

**Uses:** dims.

<b>TOOLDIR Name:</b>	const	
<b>Type:</b>	System RPL	
<b>Xlib #</b>	165	
<b>Input:</b>		
Obj	<r>	<c>
<b>Output:</b>		
ListMat		
<b>Purpose:</b>	Creates a matrix given a constant value and the number of rows and columns.	
<b>Notes:</b>	No restriction on Obj type.	
<b>Uses:</b>	->mat.	

**TOOLDIR Name:** copy

**Type:** System RPL

**Xlib #** 164

**Input:**

meta<sub>h</sub>      <f>      <l>      <d>

**Output:**

meta<sub>h</sub> (n+Abs (l-f)+1)

**Purpose:** Copies objects from <f> to <l> above <d> and updates meta-counter.

**Notes:** A number <d> greater than the meta-counter causes objects to be appended to the tail of the meta-object.

**Uses:** hdup.

**TOOLDIR Name:** delcol

**Type:** System RPL

**Xlib #** 163

**Input:**

ListMat (m,n) <col>

**Output:**

ListMat (m,n-1)

**Purpose:** Deletes specified column in ListMat.

**Notes:** Checks subscript range.

**Uses:** argerr, delete, dims, mtop.

**TOOLDIR Name:** delete

**Type:** System RPL

**Xlib #** 162

**Input:**

$\text{Obj}_1 \dots \text{Obj}_f \dots \text{Obj}_1 \dots \text{Obj}_n$  <n> <f> <l>

**Output:**

$\text{Obj}_1 \dots \text{Obj}_{f-1} \text{Obj}_{l+1} \dots \text{Obj}_n$  <n-abs(f-l+1)>

**Purpose:** Deletes objects from <f> to <l> and updates the meta-object counter.

**Notes:**  $m = n - \text{abs}(l - f + 1)$

**Uses:** hdrop.

**TOOLDIR Name:** delrow

**Type:** System RPL

**Xlib #** 161

**Input:**

ListMat (m,n) <row>

**Output:**

ListMat (m-1,n)

**Purpose:** Deletes specified row in ListMat.

**Notes:** Checks subscript range.

**Uses:** argerr, delete, dims.



**TOOLDIR Name:** determ

**Type:** System RPL

**Xlib #** 160

**Input:**

SymbMat (n)

**Output:**

Symbolic

**Purpose:** Calculates the determinant of a square matrix.

**Notes:** Does not try any simplification on the result.

**Uses:** best, mat->, metax, reduc, det2, det3, square?.

**TOOLDIR Name:** dims

**Type:** System RPL

**Xlib #** 159

**Input:**

ListMat (m,n)

**Output:**

ListMat (m,n) <m> <n>

**Purpose:** Returns Matrix dimensions. Errors out if the matrix is ill-formed.

**Notes:** {{{}} dims correctly returns <0> <0>. No check is made on object types contained in sublists. This feature allow the use of dims for purposes other than strictly symbolic matrix manipulation.

**Uses:** CHECKL, mtop.

**TOOLDIR Name:** equal?

**Type:** System RPL

**Xlib #** 158

**Input:**

ListMat<sub>1</sub> ListMat<sub>2</sub>

**Output:**

ListMat<sub>1</sub> ListMat<sub>2</sub> <m> <n> TRUE

or

Listmat<sub>1</sub> Listmat<sub>2</sub> FALSE

**Purpose:** Checks if both matrix have same dimensions.

**Notes:** none

**Uses:** dims.

**TOOLDIR Name:** factor

**Type:** System RPL

**Xlib #** 157

**Input:**

SymbMat Symbolic

**Output:**

SymbMat

**Purpose:** Multiply all elements of the matrix by the symbolic value.

**Notes:** none

**Uses:** apply.

**TOOLDIR Name:** find\$

**Type:** System RPL

**Xlib #** 156

**Input:**

String<sub>1</sub> String<sub>2</sub>

**Output:**

String<sub>1</sub> <matches>

**Purpose:** Finds occurrences of String<sub>2</sub> in String<sub>1</sub>.

**Notes:** none

**TOOLDIR Name:** det3

**Type:** System RPL

**Xlib #** 155

**Input:**  
SymbMat

**Output:**  
Symbolic

**Purpose:** Calculates the determinant of a 3x3 matrix.

**Notes:** The matrix must be of order 3 or greater. If the order is greater than 3, it calculates the principal third order minor.

**Uses:** lget.

**TOOLDIR Name:** det2

**Type:** System RPL

**Xlib #** 154

**Input:**

SymbMat

**Output:**

Symbolic

**Purpose:** Calculates the determinant of a 2x2 matrix.

**Notes:** The matrix must be of order 2 or greater. If the order is greater than 2, it calculates the principal 2nd order minor.

**Uses:** lget.

**TOOLDIR Name:** findobj

**Type:** System RPL

**Xlib #** 153

**Input:**

Comp      Obj

**Output:**

Comp      <matches>

**Purpose:** Counts how many times a given object appears within a compound object (recursively).

**Notes:** Search is limited to RAM level objects (see npos). This is necessary to prevent endless loop in recursive routines

**Uses:** npos.



**TOOLDIR Name:** gaddr

**Type:** System RPL

**Xlib #** 152

**Input:**

Grob<sub>1</sub> Grob<sub>2</sub>

**Output:**

Grob

**Purpose:** Joins Grob<sub>2</sub> to the right side of Grob<sub>1</sub>.

**Notes:** Grob<sub>1</sub> & Grob<sub>2</sub> must have same height.  
Null Grobs must be of the form 0 x h.

**TOOLDIR Name:** gaddup

**Type:** System RPL

**Xlib #** 151

**Input:**

Grob<sub>1</sub>      Grob<sub>2</sub>

**Output:**

Grob

**Purpose:** Joins Grob<sub>2</sub> at the top of Grob<sub>1</sub>.

**Notes:** Grob<sub>1</sub> & Grob<sub>2</sub> must have same widths.  
Null Grobs must look like Graphic w x 0.

**TOOLDIR Name:** hdrop

**Type:** System RPL

**Xlib #** 150

**Input:**

$\text{Obj}_n \dots \text{Obj}_1$        $\langle f \rangle$        $\langle l \rangle$

**Output:**

$\text{Obj}_n \dots \text{Obj}_f \text{Obj}_1 \dots \text{Obj}_1$

**Purpose:** Deletes levels from  $\langle f \rangle$  to  $\langle l \rangle$ .

**Notes:** none

**Uses:** rdown.

**TOOLDIR Name:** hdup

**Type:** System RPL

**Xlib #** 149

**Input:**

$\text{Obj}_n \dots \text{Obj}_f \dots \text{Obj}_1 \dots \text{Obj}_d \dots \text{Obj}_1$  <f> <l> <d>

**Output:**

$\text{Obj}_n \dots \text{Obj}_f \dots \text{Obj}_1 \dots \text{Obj}_f \dots \text{Obj}_1 \text{Obj}_d \dots \text{Obj}_1$

**Purpose:** Copies levels between <f> and <l> above level <d>.

**Notes:** none.

**Uses:** hshift, rdown, rup.

**TOOLDIR Name:** hshift

**Type:** System RPL

**Xlib #** 148

**Input:**

Obj<sub>1</sub>      ...      Obj<sub>n</sub>      <f>      <l>      <d>

**Output:**

Obj<sub>1</sub>      ...      Obj<sub>n</sub>

**Purpose:** Moves levels between <f> and <l> above level <d>.

**Notes:** none

**Uses:** MDS, MUS.

**TOOLDIR Name:** idn

**Type:** System RPL

**Xlib #** 147

**Input:**

<n>

**Output:**

Symbmat(n)

**Purpose:** Creates a n by n Identity Matrix.

**Notes:** none

**Uses:** ->mat.

**TOOLDIR Name:** idx?

**Type:** System RPL

**Xlib #** 146

**Input:**

Obj<sub>1</sub>      Obj<sub>2</sub>

or

List      Obj<sub>2</sub>

**Output:**

or

'idx'

**Purpose:** Returns nothing if Obj<sub>1</sub> and Obj<sub>2</sub> are equal or if Obj<sub>2</sub> is contained in List. Otherwise returns 'idx'

**Notes:** none

**TOOLDIR Name:** keywait

**Type:** System RPL

**Xlib #** 145

**Input:** Keyboard

**Output:**

<key#> <shift>

**Purpose:** Returns the code of the key being pressed.

**Notes:** none.



**TOOLDIR Name:** 12m

**Type:** System RPL

**Xlib #** 144

**Input:**

n : Obj  
...: ...  
m+2: Obj  
m+1: 'mark  
m : Obj  
...: Obj  
1 : Obj

**Output:**

n-m: Obj  
... : ...  
2 : Obj  
1 : List

**Purpose:** Groups together objects between TOS and first private mark.

**Notes:** The mark is 'mark.  
Same as DEPTH ->LIST if mark not found. The marker is always removed.

**Uses:** c2m.

**TOOLDIR Name:** lget

**Type:** System RPL

**Xlib #** 143

**Input:**

ListMat <m> <n>

**Output:**

Object

**Purpose:** Gets element (m,n) from ListMat.

**Notes:** Newob is always performed on the element.

**Uses:** argerr.

**TOOLDIR Name:** lines->

**Type:** System RPL

**Xlib #** 142

**Input:**

meta<sub>h</sub>

**Output:**

Str

**Purpose:** Joins objects by mean of an LF character.

**Notes:** Normally, object types should be in range 0-19 (User Objects).

**TOOLDIR Name:** log2

**Type:** System RPL

**Xlib #** 141

**Input:**

<h>

**Output:**

<b>

**Purpose:** Returns the position of the most significant bit of <h>.

**Notes:** <0> means that  $h=0$  (no bit set to 1). <2> means  $2^1$  bit set, <3> means  $2^2$  bit set and so on.

**TOOLDIR Name:** lop1

**Type:** System RPL

**Xlib #** 140

**Input:**

{ Obj ... Obj } { Cmd ... Cmd }

**Output:**

{ Obj ... Obj }

**Purpose:** Applies commands in list to all elements of list in level 2 and puts the result in a new list. The final list should contain the same number of objects than the original one.

**Notes:** none.

**Uses:** mtop.

**TOOLDIR Name:** lopn

**Type:** System RPL

**Xlib #** 139

**Input:**

{ Obj ... Obj } { Cmd ... Cmd }

**Output:**

{ { Obj ... Obj } ... { Obj ... Obj } }

**Purpose:** Applies commands in list to all elements of list in level 2 and puts the result in a new list. The final list can contain a different number of elements than the original one.

**Notes:** Every command in the list must be properly setup to work on a list.

**Uses:** l2m, mark.

**TOOLDIR Name:** lput

**Type:** System RPL

**Xlib #** 138

**Input:**

Listmat    <m>            <n>            Obj

**Output:**

Symbmat

**Purpose:**            Puts Element (m,n) in the Matrix.

**Notes:**            none.

**Uses:**             argerr.

**TOOLDIR Name:** lvop

**Type:** System RPL

**Xlib #** 137

**Input:**

{Obj ... Obj} {Obj ... Obj} {Cmd ... Cmd}

**Output:**

{Obj ... Obj}

**Purpose:** Applies commands to each pair of object and puts the result in a new list. Occurences of symbol 'idx' are substituted by the current counter value.

**Notes:** none.

**Uses:** BIND, l2m, mark.



**TOOLDIR Name:** mark

**Type:** Global Name

**Xlib #** 136

**Input:** none

**Output:**

'mark

**Purpose:** Pushes on the stack the private mark.

**Notes:** This marker is different from user MARK. If you counts up to mark with 'MARK on the stack, c2m ignores it and counts up to the first 'mark (if any).

**TOOLDIR Name:** mat->

**Type:** System RPL

**Xlib #** 135

**Input:**

Symbmat

**Output:**

$\text{Elm}_1 \quad \dots \quad \text{Elm}_{m \times n} \quad \langle m \rangle \langle n \rangle$

**Purpose:** Splits Symbmat on the stack.

**Notes:** none.

**Uses:** dims.

**TOOLDIR Name:** metop

**Type:** System RPL

**Xlib #** 134

**Input:**

Meta {Cmd ... Cmd}

**Output:**

Meta

**Purpose:** Applies each command to all elements of meta-object. The sequence of commands must not alter the size of the meta-object.

**Notes:** Meta-object is scrolled upwards. A check is performed to ensure enough objects are on the stack

**Uses:** mtop.

**TOOLDIR Name:** metopn

**Type:** System RPL

**Xlib #** 133

**Input:**

meta {Cmd ... Cmd}

**Output:** none

meta(list)

**Purpose:** Applies Commands to all elements of meta-object handling multiple results. Results of the command list are grouped in a list. The final meta-object is composed of lists retaining the original meta-object size.

**Notes:** Meta-object is scrolled upwards. A check is performed to ensure enough objects are on the stack.

**Uses:** mtopn.

**TOOLDIR Name:** move

**Type:** System RPL

**Xlib #** 132

**Input:**

$\text{Obj}_1 \dots \text{Obj}_f \dots \text{Obj}_l \dots \text{Obj}_d \dots \text{Obj}_n$  <n> <f> <l> <d>

**Output:**

$\text{Obj}_1 \dots \text{Obj}_{f-1} \text{Obj}_{l+1} \dots \text{Obj}_f \dots \text{Obj}_l \text{Obj}_d \dots \text{Obj}_n$  <n>

**Purpose:** Moves Objects from f to l to the new position above d.

**Notes:** a value greater than n means the tail of the meta. f and l are not required to be ordered.

**Uses:** hshift.

**TOOLDIR Name:** mrev

**Type:** Code

**Xlib #** 131

**Input:**

metah

**Output:**

metah

**Purpose:** Reverses the order of the meta-object.

**Notes:** none.

**TOOLDIR Name:** msymb?

**Type:** System RPL

**Xlib #** 130

**Input:**

Symbmat

**Output:**

Symbmat TRUE

or

pseudo-meta FALSE

**Purpose:** Checks if Symbmat is a valid Symbolic matrix.

**Notes:** Illegal items are tagged with the string "Syntax".

**Uses:** mat->, mtop, ->mat.

**TOOLDIR Name:** mtop

**Type:** System RPL

**Xlib #** 129

**Input:**

metah { cmd ... cmd }

**Output:**

metah

**Purpose:** Applies the command list to each element in the meta-object. The sequence of commands must not alter the size of the meta-object, i.e. it must return a single object.

**Notes:** none.



**TOOLDIR Name:** mtopn

**Type:** System RPL

**Xlib #** 128

**Input:**

metah { cmd ... cmd }

**Output:**

metah(lists)

**Purpose:** Applies the command list to all elements of the meta-object, handling multiple results. Results are grouped in a list. The final meta-object is composed of lists and retains the original meta-object size.

**Notes:** none.

**Uses:** l2m, mark.

**TOOLDIR Name:** mult

**Type:** System RPL

**Xlib #** 127

**Input:**

Symbmat1 Symbmat2

**Output:**

Symbmat

**Purpose:** Row by Column Matrix Multiplication.

**Notes:** Issues <33501> error if conform? returns FALSE.

**Uses:** conform?, lget.

<b>TOOLDIR Name:</b>	ndupn	
<b>Type:</b>	System RPL	
<b>Xlib #</b>	126	
<b>Input:</b>	Obj            <n>	
<b>Output:</b>		
Obj            ...	Obj	<n>
<b>Purpose:</b>	Duplicates Obj <n-1> times and leaves <n> on stack, i.e. creates a meta-object composed of n copies of Obj.	
<b>Notes:</b>	On <n>=0 a null meta is created. Same as Entry Point #5E370	

**TOOLDIR Name:** null

**Type:** System RPL

**Xlib #** 125

**Input:**

Obj

**Output:**

NullObj

**Purpose:** Replaces Obj with the corresponding null object (when applicable).

**Notes:** Table of null objects:

Real	0
Complex	(0,0)
Binary Integer	null(#)
System Binary	<0>
Value_Unit	Zero_Unit
Real Array (m,n)	Null_Real Array (m,n)
Complex Array (m,n)	Null Complex Array (m,n)
Symbolic	0
Global	0
Local	0
String	""
List	{ }
Grob (w x h)	Grob (0 x h)

**TOOLDIR Name:** pkmeta

**Type:** System RPL

**Xlib #** 124

**Input:**

$\text{Obj}_1 \dots \text{Obj}_n$   $\langle n \rangle$   $\langle o \rangle$   $\langle c \rangle$   $\langle r \rangle$   $\langle h \rangle$

**Output:**

$\text{meta}_h$   $\langle o \rangle$   $\langle c \rangle$   $\text{Obj}_{o+c}$  1

or

$\text{meta}_h$   $\langle o \rangle$   $\langle c \rangle$  0

**Purpose:** Interactive selection of objects.

**Notes:**  $\langle o \rangle + 1$  is the number of the first element in the window.  
 $\langle c \rangle$  is the number of current element in the window.  
 $\langle o + c \rangle$  is the absolute current element.  
 $\langle r \rangle$  is the number of rows of the window (typically  $\langle 5 \rangle$ ).  
 $\langle h \rangle$  is the row of the display from which starts the window.

To visualize a 7 levels window is necessary to disable the ticking clock.

**Uses:** ACTIONS, ENH, KEYS, metop.

**TOOLDIR Name:** rdown

**Type:** System RPL

**Xlib #** 123

**Input:**

$\text{Obj}_1 \dots \text{Obj}_t \dots \text{Obj}_n$        $\langle n \rangle$        $\langle t \rangle$

**Output:**

$\text{Obj}_{n-t+1} \dots \text{Obj}_1 \dots \text{Obj}_{n-t}$

**Purpose:** rolls down n objects t times.

**Notes:** none.

**Uses:** RD, RU.

**TOOLDIR Name:** rdrop

**Type:** System RPL

**Xlib #** 122

**Input:**

Obj<sub>1</sub>      ...      Obj<sub>n</sub>      <f>      <l>

**Output:**

Obj<sub>1</sub>      ...      Obj<sub>n</sub>      <n-l+f-1>

**Purpose:** Deletes objects from level f to level l.

**Notes:** none.

**Uses:** hdrop.

**TOOLDIR Name:** rdup

**Type:** System RPL

**Xlib #** 121

**Input:**

$\text{Obj}_n \dots \text{Obj}_1$        $\langle f \rangle$        $\langle l \rangle$        $\langle d \rangle$

**Output:**

$\text{Obj}_n \dots \text{Obj}_{d-1} \text{Obj}_f \dots \text{Obj}_1 \text{Obj}_d \dots \text{Obj}_1$

**Purpose:** Copies objects from level f to level l above level d.

**Notes:** a value 0 for d means the top of the stack.

**Uses:** hdup.



**TOOLDIR Name:** replace

**Type:** System RPL

**Xlib #** 120

**Input:**

Comp      obj<sub>f</sub>      obj<sub>r</sub>      <0>

**Output:**

Comp      <repld>

**Purpose:** Replaces all the occurrences of obj<sub>f</sub> with obj<sub>r</sub> within the compound object.

**Notes:** <repld> counts the replacements.

**Uses:** TifC.

**TOOLDIR Name:** replace\$

**Type:** System RPL

**Xlib #** 119

**Input:**

String1 String2 String3

**Output:**

String <repld>

**Purpose:** Replaces all the occurrences of string2 in String1 with String3.

**Notes:** <repld> counts the replacements.

**Uses:** splith.

**TOOLDIR Name:**           reduc

**Type:**                   System RPL

**Xlib #**                   118

**Input:**

SymbMat    <j>

**Output:**

Symbmat(n-1)

**Purpose:**                Reduces the matrix to n-1 order.

**Notes:**

```
for k=j+1 to n
  for i=2 to n
    mat(i,k)=mat(i,k)-mat(1,k)*mat(i,j)/mat(1,j)
  next i
next k
newmat=cmpl(mat,1,j)
```

**Uses:**                cmpl, lget, lput.

**TOOLDIR Name:** Todd

**Type:** Code

**Xlib #** 117

**Input:**

<n>

**Output:**

T/F

**Purpose:** Returns TRUE if n is odd.

**Notes:** none.

**TOOLDIR Name:** dot

**Type:** System RPL

**Xlib #** 116

**Input:**

ListMat1 (m,n)      ListMat2 (m,n)      prg

**Output:**

ListMat

**Purpose:** Applies the binary operator prg to all the pairs of elements and puts the result in a new matrix.

**Notes:** Matrices must have same dimension. prg must be a procedure taking two elements from the stack and returning a single result.

**Uses:** equal?, lget.

**TOOLDIR Name:** rowcol

**Type:** System RPL

**Xlib #** 115

**Input:**

Str <pos>

**Output:**

String <Row> <Col>

**Purpose:** Returns the Row and Column position of given absolute location (with respect to Linefeeds).

**Notes:** none.

**TOOLDIR Name:** rpt

**Type:** System RPL

**Xlib #** 114

**Input:**

List <n>

or

String <n>

or

Grob <n>

**Output:**

List

or

String

or

Grob

**Purpose:** Creates a new object composed of n repetitions of the object given.

**Notes:** none.

**Uses:** addobj, log2, null.

**TOOLDIR Name:** rptcmd

**Type:** System RPL

**Xlib #** 113

**Input:**

... Obj Prg <n>

or

... Obj Global <n>

or

... Obj Local <n>

**Output:**

Any

**Purpose:** Repeats the program n times.

**Notes:** This routine allows to execute repeatedly some trivial tasks, without sacrificing any speed. It allows also some nice tricks like this ultra-short routine for finding the maximum real number in a list):

Comp->	#054AF
Dup	#03188
T_If_h>1	#636F0
If_F_Doerr_Bad_Arg_Val	#63B19
h=h-1	#03E0E
Push_Next	#06E97
R=Max(R2,R1)	#2A6F5
Swap	#03223
rptcmd	XLIB 821 113

rptcmd neither traps errors nor checks stack overflows.



**TOOLDIR Name:** rup

**Type:** System RPL

**Xlib #** 112

**Input:**

$\text{Obj}_1 \dots \text{Obj}_n$        $\langle n \rangle$        $\langle t \rangle$

**Output:**

$\text{Obj}_{t+1} \dots \text{Obj}_n$      $\text{Obj}_1 \dots \text{Obj}_t$        $\langle n \rangle$

**Purpose:** Rolls upwards  $\langle n \rangle$  Objects  $\langle t \rangle$  times.

**Notes:** The shortest scroll direction is chosen automatically.  
Rolling up 100 Objects 99 times isn't smart at all!.

**Uses:** RD, RU.

**TOOLDIR Name:** shift

**Type:** System RPL

**Xlib #** 111

**Input:**

$\text{Obj}_{\text{MAX}(f, l, d)} \dots \text{Obj}_1$        $\langle f \rangle$        $\langle l \rangle$        $\langle d \rangle$

**Output:**

$\text{Obj}_{\text{MAX}(f, l, d)-1} \dots \text{Obj}_f \dots \text{Obj}_1$      $\text{Obj}_d \dots \text{Obj}_1$      $\langle n \rangle$

**Purpose:** Moves Objects from level  $f$  through level  $l$  to the new position above level  $d$ .

**Notes:**  $d=\langle 0 \rangle$  means the TOS.

**Uses:** hshift.

**TOOLDIR Name:** sortany

**Type:** System RPL

**Xlib #** 110

**Input:**

Obj<sub>1</sub>      ...      Obj<sub>n</sub>      <n>      Prg

**Output:**

Obj<sub>s1</sub>      ...      Obj<sub>sn</sub>      <n>

**Purpose:** Sorts Objects using the sort procedure in level 1. The procedure must take two elements from the stack and return a TRUE or FALSE value.

**Notes:** To sort in ascending order, the procedure must perform a ">" comparison. For example, to sort in ascending order Tagged Reals, the predicate could be :

```
Begin
  DTAG
  Swap          #03223
  DTAG
  Swap          #03223
  T_If_R2>R1    #2A88A
End
```

or even better...

```
Begin
  DTAG
  Swap          #03223
  DTAG
  T_If_R2<R1    #2A871
End
```

**Uses:** xlev.

**TOOLDIR Name:** findrow

**Type:** System RPL

**Xlib #** 109

**Input:**

SymbMat

**Output:**

<r> TRUE

FALSE

**Purpose:** Finds the best row in the matrix on which performing determinant's calculation.

**Notes:** none.

**Uses:** mtop, rptcmd, weight.

**TOOLDIR Name:** best

**Type:** System RPL

**Xlib #** 108

**Input:**

SymbMat

**Output:**

SymbMat    <r>        <c>

**Purpose:** Finds the best element in the matrix on which performing calculations.

**Notes:** none.

**Uses:** findrow, lget, srccol.

**TOOLDIR Name:** search

**Type:** System RPL

**Xlib #** 107

**Input:**

"search\_key" <occur> Global

**Output:**

**Purpose:** Positions the cursor on the <occur>-th match of the search key in the editor.

**Notes:** <occur>=<0> means last occurrence.

**TOOLDIR Name:** weight

**Type:** System RPL

**Xlib #** 106

**Input:**

**Output:**

**Purpose:** Weights expressions by object type. The coefficients follow:

Real:	<1> ( <sup>3</sup> )
Complex:	<2>
Global:	<6>
Local:	<6>
Algebraic:	<A>
Unit:	<E>

**Notes:** (<sup>3</sup>) a Real value of 0 weights <0>.

**TOOLDIR Name:** splith

**Type:** System RPL

**Xlib #** 105

**Input:**

String <p>

**Output:**

String(1,p-1)      String(p,p)      String(p+1)

**Purpose:** Splits the string in three chunks.

**Notes:** <p> is set to min(p,maxlen). Null strings are returned when necessary.  
<p> must be greater than <0>.



**TOOLDIR Name:** splitl

**Type:** System RPL

**Xlib #** 104

**Input:**

$\{ \text{Obj}_1 \dots \text{Obj}_{p-1} \text{Obj}_p \text{Obj}_{p+1} \dots \text{Obj}_n \}$   $\langle p \rangle$

**Output:**

$\{ \text{Obj}_1 \dots \text{Obj}_{p-1} \}$   $\text{Obj}_p$   $\{ \text{Obj}_{p+1} \dots \text{Obj}_n \}$

**Purpose:** Splits the list in three chunks.

**Notes:**  $\langle p \rangle$  is set to  $\min(p, \text{maxlen})$ .  $\langle p \rangle$  must be greater than  $\langle 0 \rangle$ .  
The command works also for Program Objects and for Algebraics.

**TOOLDIR Name:** square?

**Type:** System RPL

**Xlib #** 103

**Input:**

SymbMat

**Output:**

SymbMat <n> TRUE

SymbMat FALSE

**Purpose:** Checks for a square matrix.

**Notes:** none

**Uses:** dims.

**TOOLDIR Name:** srccol

**Type:** System RPL

**Xlib #** 102

**Input:**

{ Obj<sub>1</sub> ... Obj<sub>n</sub> }

**Output:**

<c> TRUE

FALSE

**Purpose:** Finds the first element not equal to 0.

**Notes:** none

**TOOLDIR Name:**           subt

**Type:**                   System RPL

**Xlib #**                   101

**Input:**

SymbMat1 (m,n)           SymbMat2 (m,n)

**Output:**

SymbMat (m,n)

**Purpose:**               Subtracts SymbMat2 from SymbMat1 element by element.

**Uses:**                 dot.

**TOOLDIR Name:** trn

**Type:** System RPL

**Xlib #** 100

**Input:**

SymbMat (m,n)

**Output:**

SymbMat (n,m)

**Purpose:** Transposes the Matrix.

**Uses:** dims, getcol.

**TOOLDIR Name:** xlvls

**Type:** System RPL

**Xlib #** 99

**Input:**

N+2: Obj<sub>n</sub>  
...:  
L+2: Obj<sub>1</sub>  
...:  
K+2: Obj<sub>k</sub>  
...:  
3 : Obj<sub>1</sub>  
2 : <l>  
1 : <k>

**Output:**

N : Obj<sub>n</sub>  
...:  
L : Obj<sub>k</sub>  
:  
K : Obj<sub>1</sub>  
:  
1 : Obj<sub>1</sub>

**Purpose:** Swaps pointers of object k and l.

**Notes:** Both stack depth and zero values are checked.

**Uses:** xlev.

**TOOLDIR Name:** ->lines

**Type:** System RPL

**Xlib #** 98

**Input:**

Str

**Output:**

Str<sub>1</sub>      ...      Str<sub>n</sub>      <n>

**Purpose:** Splits the string in several lines truncating it at Linefeeds.

**Notes:** LF characters embedded in double quotes are ignored. After each subdivision the linefeed is deleted. The inverse function is lines-> on page 131.

**Uses:** LFPOS.

**TOOLDIR Name:** ->mat

**Type:** System RPL

**Xlib #** 97

**Input:**

Obj1 ... Objn <m> <n>

**Output:**

ListMat(m,n)

**Purpose:** Builds up a list of list given a pseudo meta-object.

**Notes:** Doesn't allow a 0 value for m or n (issues BadArgV).



**TOOLDIR Name:** xlev  
**Type:** Code  
**Xlib #** 96

**Input:**

N+2: Obj<sub>n</sub>  
...:  
L+2: Obj<sub>1</sub>  
...:  
K+2: Obj<sub>k</sub>  
...:  
3 : Obj<sub>1</sub>  
2 : <l>  
1 : <k>

**Output:**

N : Obj<sub>n</sub>  
...:  
L : Obj<sub>k</sub>  
:  
K : Obj<sub>1</sub>  
:  
1 : Obj<sub>1</sub>

**Purpose:** Swaps pointers of object l and k.

**Notes:** No check is made on stack pointers.



## Index

->torf 14

### A

add 98  
addcon 99  
addobj 100  
addp 40  
apply 68  
argerr 101

### B

best 165  
bind 82  
box 33  
bsearch 10

### C

c2m 102  
center 23  
checkl 83  
chlq 103  
chsetq 104  
chstq 105  
ckl2r 85  
ckobj 84  
ckr 76  
ckrol 77  
cmpl 106  
config 81  
conformq 107  
const 108  
copy 109

### D

delcol 110  
delete 111  
delrow 112  
det2 119  
det3 118  
determ 113  
diff 27  
dims 114  
dot 157

### E

equalq 115  
expbuf 24

### F

factor 116  
fill 45  
findobj 120  
findrow 164  
finds 117

### G

gaddr 121  
gaddup 122  
getaddr 63  
getcol 73  
gop 34

### H

hdrop 123  
hdup 124  
hshift 125

### I

idn 126  
idxq 127  
inter 28

### K

k 79  
keywait 128  
ktos 78

### L

l2m 129  
lfpos 87  
lget 130  
line 35  
linesto 131  
linetypes 30  
log2 132  
lop1 133  
lopn 134  
lput 135  
ltrim 48  
lvop 136  
lwc 49

### M

ma2 71  
madd1 9  
mark 127

matto 138  
mcentr 22  
mds 88  
member 59  
metax 72  
metop 139  
metopn 140  
move 141  
mrev 142  
msymbq 143  
mtop 144  
mtopn 145  
mult 146  
mus 91

## **N**

ncount 62  
ndupn 147  
nget 67  
norm 51  
npos 61  
nput 66  
null 148

## **O**

ord 38

## **P**

pad\$ 13  
patterns 41  
pkmeta 149  
polygon 19  
popp 39  
PPARdef 21  
putobj 65

## **R**

rd 92  
rdown 150  
rdrop 151  
rdup 152  
rect 37  
red 26  
reduc 155  
replace 153  
replaces 154  
revb 54  
revs 55  
revsys 70

rowcol 158  
rpt 159  
rptcmd 160  
rtrim 47  
ru 93  
rup 161

## **S**

scan 60  
scanp 25  
search 166  
shift 162  
sort 12  
sortany 163  
span 58  
splith 168  
splitl 169  
splito 17  
splits 18  
squareq 170  
srccol 171  
subt 172

## **T**

tifc 64  
todd 156  
toext 69  
tolines 175  
tomat 176  
tomsg 53  
toprolog 57  
trim 52  
trn 173

## **U**

union 29  
upc 50

## **V**

vfill 46  
visit 75

## **W**

weight 167

## **X**

xlev 177  
xlibto 56  
xlvl 174

## Contents

Notice	1
FOREWORD . . . . .	1
Legenda . . . . .	2
Notes . . . . .	2
Commands sorted by category . . . . .	2
String Utilities . . . . .	3
Binary string Utilities . . . . .	3
System Binary Utilities . . . . .	4
List Manipulation Utilities . . . . .	4
General purpose Utilities . . . . .	4
Program editing utilities . . . . .	5
Stack Manipulation Utilities . . . . .	5
Meta-object Manipulation Utilities . . . . .	5
Symbolic Math Utilities . . . . .	6
Type Conversion Utilities . . . . .	6
Graphics Utilities . . . . .	7
Object Manipulation Utilities . . . . .	7
Configuration . . . . .	7
Command Reference	9
Index	179