

HP48G Entry Reference

Complete listing sorted by functionality
Edition 2.11, 30 May 2005

Carsten Dominik, Thomas Rast & Eduardo M. Kalinowski

Table of Contents

1	Introduction	1
1.1	Disclaimer and Acknowledgments	1
1.2	Terminology	3
1.2.1	Abbreviations used in Stack Diagrams	3
1.2.2	Unsupported Entry Points	3
1.2.3	More Information	4
2	HP Objects	5
2.1	Binary Integers	5
2.1.1	Built-in BINTS 0-127	5
2.1.2	Built-in BINTS 127-255	9
2.1.3	Built-in BINTS 256-	10
2.1.4	Pushing Several BINTs	12
2.1.5	Conversion	13
2.1.6	Arithmetic Functions	14
2.1.7	Tests	16
2.2	Real Numbers	16
2.2.1	Built-in Real Numbers	17
2.2.2	Built-in Extended Real Numbers	18
2.2.3	Stack Manipulation Combined with Reals	19
2.2.4	Conversion	19
2.2.5	Real Functions	20
2.2.6	Extended Real Functions	22
2.2.7	Tests	23
2.3	Complex Numbers	24
2.3.1	Built-in Complex Numbers	24
2.3.2	Conversion	24
2.3.3	Functions	24
2.3.4	Tests	26
2.3.5	Real and Complex Number Functions	26
2.4	Character Strings	27
2.4.1	Built-in Characters	27
2.4.2	Built-in Strings	29
2.4.3	Built-in Strings with Stack Manipulation	32
2.4.4	Conversion	32
2.4.5	Management	32
2.4.6	Parsing Strings	35
2.4.7	Decompilation	36
2.4.8	String Tests	37
2.5	HEX Strings	38
2.5.1	Conversion	38
2.5.2	General Functions	38
2.5.3	Tests	40

2.6	Tagged Objects	40
2.7	Arrays	40
	2.7.1 General Functions	41
	2.7.2 Conversion	44
	2.7.3 Statistics	44
2.8	Unit Objects	45
	2.8.1 Creating Units	45
	2.8.2 General Functions	46
	2.8.3 Arithmetic Functions	46
	2.8.4 Tests	47
2.9	Composites	47
	2.9.1 General Operations	48
	2.9.2 Building	50
	2.9.3 Exploding	51
	2.9.4 Lists	51
	2.9.5 Secondaries	52
2.10	Meta Objects	52
	2.10.1 Stack Functions	52
	2.10.2 Combining Functions	53
	2.10.3 Meta and Object Operations	53
	2.10.4 Other Operations	55
2.11	Symbolics	56
	2.11.1 General Operations	56
	2.11.2 Mathematical Operations	59
	2.11.3 Collection	63
	2.11.4 Expansion	64
	2.11.5 Integration	65
	2.11.6 Derivatives	66
	2.11.7 Other Functions	67
	2.11.8 Meta Symbolics Functions	68
2.12	Library and Backup Objects	72
	2.12.1 Port Operations	72
	2.12.2 Rompointers	72
	2.12.3 Libraries	73
	2.12.4 Backup Objects	75

3	General SysRPL Entries	76
3.1	Stack Operations	76
3.2	Temporary Environments	79
3.2.1	Built-in IDs and LAMs	79
3.2.2	Conversion	80
3.2.3	Temporary Environments Words	80
3.3	Error Handling	83
3.3.1	General Words	83
3.3.2	Error Generating Words	85
3.4	Conditionals	87
3.4.1	Boolean Flags	87
3.4.2	General Tests	88
3.4.3	True/False Tests	89
3.4.4	Binary Integer Tests	91
3.4.5	Real and Complex Number Tests	92
3.4.6	Meta Object Tests	93
3.4.7	General Object Tests	94
3.4.8	Miscellaneous	95
3.5	Runstream Control	96
3.5.1	Quoting Objects	98
3.5.2	Skipping Objects	99
3.6	Loops	100
3.6.1	Indefinite Loops	100
3.6.2	Definite Loops	100
3.7	Memory Operations	102
3.7.1	Recalling, Storing and Purging	102
3.7.2	Directories	103
3.7.3	The Hidden Directory	105
3.7.4	Temporary Memory	106
3.8	Time and Alarms	106
3.8.1	Alarms	108
3.9	System Functions	109
3.9.1	User and System Flags	110
3.9.2	General Functions	112
3.10	Kermit	114

4	Input and Output	119
4.1	Checking for Arguments	119
4.1.1	Number and Type of Arguments	119
4.1.2	Type Checking	121
4.2	Keyboard Control	123
4.2.1	Converting Keycodes	123
4.2.2	Waiting for Keys	124
4.2.3	The ATTN Flag	125
4.2.4	Bad Keys	126
4.2.5	User Keys	126
4.3	The Menu	127
4.3.1	Menu Properties	127
4.3.2	Building Menus	129
4.3.3	Menu Display	130
4.3.4	Displaying Menu Labels	131
4.3.5	General Entries	131
4.4	InputLine and Inputforms	132
4.4.1	Inputline	132
4.4.2	Inputform	132
4.5	The Browser Engines	132
4.5.1	The HP48 Browser Engine	132
4.5.2	The HP49 Browser Engine	136
4.6	The Parametrized Outer Loop (POL)	136
4.7	Editor Commands	137
4.7.1	Status	137
4.7.2	Display Window	138
4.7.3	Inserting Text	138
4.7.4	Moving the Cursor	139
4.7.5	Starting the Editor	139
4.7.6	Miscellaneous	140
4.8	Entries Related to the Matrix Editor and Matrix Operations	141
4.9	The Display	141
4.9.1	Display Organization	141
4.9.2	Preparing the Display	142
4.9.3	Immediate Refresh	143
4.9.4	Controlling Display Refresh	143
4.9.5	Clearing the Display	147
4.9.6	Annunciator and Modes Control	147
4.9.7	Window Coordinates	148
4.9.8	Scrolling the Display	149
4.9.9	Displaying Objects	150
4.9.10	Displaying Text	150
4.9.11	Messages and Boxes	151
4.10	Graphics	152
4.10.1	Built-in Grobs	152
4.10.2	Dimensions	152
4.10.3	Grob Handling	153

4.10.4	Creating Menu Label Grobs	154
4.10.5	Converting Strings to Grobs	156
4.11	Plotting	156
5	The HP49G CAS	161
5.1	Matrix Operations	161
5.1.1	Other Matrix Operations	161
6	UserRPL Commands	162
6.1	A-F	162
6.2	G-M	192
6.3	N-S	208
6.4	T-Z	234
6.5	Non A-Z	247
7	ML Entry Points	256
7.1	General Purpose	256
7.2	Errors	256
7.2.1	Generating Errors	256
7.2.2	Error Number Constants	256
7.3	Hexadecimal Math	257
7.4	Long Reals	257
7.4.1	Storage Handling	257
7.4.2	Calculating	258
7.4.3	Conversion	258
7.5	Memory Handling	258
7.5.1	General Memory Handling Routines	258
7.5.2	Moving and Swapping Memory Areas	259
7.5.3	Allocating Memory in TEMPOB	259
7.5.4	Resizing TEMPOB Areas	260
7.5.5	CRC Routines	260
7.5.6	Working with Memory	261
7.5.7	Other Routines	261
7.6	Memory Addresses	261
7.7	Display	262
7.8	Popping and Pushing	262
7.8.1	Pointers	262
7.8.2	TRUE and FALSE	263
7.8.3	System Binary Integers (BINT)	263
7.8.4	HXS and ZINTs	264
7.8.5	Real and Complex Numbers	264
7.9	Keyboard Handling	264
7.10	Various ML Entries	266
7.11	Object Types	266

8	RAM entries	270
8.1	RPL pointers	270
8.2	Memory management pointers	270
8.3	Screen related	270
8.4	Annunciators	270
8.5	Save areas	271
8.6	System and User Flags	271
8.7	Internal System Flags	272
8.8	Warmstart log	275
8.9	Command line management	275
8.10	POL variables	275
8.11	UART buffering	276
8.12	ROM Part Tables	276
8.13	Constants	276
8.14	Other/Uncategorized	276
9	Miscellaneous Entries	282
9.1	Undescribed Entry Points	282
10	Entries sorted by address	290
	Entry Index	331

1 Introduction

This is a list of SystemRPL, User RPL and ML entries. The list groups the entries by task in many different chapters and sections. If you are looking for a particular entry go directly to the Index. There is also an address-sorted list, if you want to look up a particular address.

1.1 Disclaimer and Acknowledgments

The information provided in this document was compiled from a large variety of sources. The transformation of all the different formats to a single database was largely done with special purpose programs to reverse-engineer the different documents. This has worked very well in many cases, and less well in some other cases. If some of the information looks oddly formatted, the reason is probably the automatic extraction.

Many of the authors of the original documents will find literal bits and pieces of their text in this document. Thanks to all of them for their generosity in allowing me to use their documents and files freely.

Neither we nor the authors of the different sources assume any warranty. This document is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

If you find any errors, let us know so that the database can be updated and fixed. Sent bug reports and other comments to [Carsten Dominik](#). Reports about the ML chapter should be sent directly to [Thomas Rast](#), but with a CC to [Carsten Dominik](#).

Here is a list of sources which have been used.

Programming in System RPL by Eduardo Kalinowski

This book has been a major source for the database. The entire book has been reverse-engineered using pdftotext and then a variety of Emacs and Perl programs to extract and format the reference part of the book.

CAS Documentation Draft by Bernard Parisse

Bernard Parisse has kindly sent me a file with draft documentation about most CAS entries which is the basis of the CAS chapter. This covers both code derived from Erable (written by Bernard) and from ALG48 (by Mika Heiskanen and Claude-Nicolas Fiechter). The documentation is not complete, and not entirely up-to-date. However, the information given should be accurate.

entries.srt by Mika Heiskanen

Mika's really useful collection of entry description has been used to double-check the information derived from Eduardo's book.

ML entry descriptions by Peter Geelhoed

Peter Geelhoed created the initial version of the ML section for this document.

HP48/49 entry cross-reference by Joe Horn

This document has been used to make a list of entries for the HP49 in the first place, and to add and double-check addresses for both calculators.

Various posts on `comp.sys.hp48`

A number of post on `comp.sys.hp48` have documented a set of entry points, for example the Graphical Toolbox (Cyrille de Brebisson), the Editor related entries (myself) and other stuff.

Supported entry lists from HP

HP has published lists of supported entries for all calculators in the database. The lists generally only contain names and addresses, no further description.

Further contributions

Denis Martinez, Alberto Zamora Oyarce, Wolfgang Rautenberg, Michael de Coninck, Christoph Giesselink, Martin Lang, Piotr Kowalewski, Lilian Pigallio and in particular Jean-Yves Avenard have also contributed information about various entry points and/or have replied to my questions about different aspects related to entries.

1.2 Terminology

1.2.1 Abbreviations used in Stack Diagrams

Here is a list of the codes use to denote different objects in the stack diagrams.

ob	any object
1...n	n objects
#	binary integer (BINT)
HXS	hex string (User binary integer)
CHR	character
\$	character string
T	TRUE
F	FALSE
flag	TRUE or FALSE
%	real number
%%	extended real number
%C	complex number
%%C	extended complex number
z, Z ,ZINT	infinite precision integer
N	positive infinite precision integer
s, symb	symbolic
u, unit	unit object
{}	list
A, []	Array
V, []	Vector
M, [[]]	Matrix
P	Polynom, a list of Qs
Q	ZINT or P
meta, ob1..obn #n	meta object
grob	graphical object
menu	list or program returning a list

UserRPL stack diagrams use some additional abbreviations

x,y	real, list, generic UserRPL object
c, (,)	complex number
#	hex string (User binary integer)
greek theta	angle (a real number)
m,n	integer (ZINT or real)
date	DD.MMYYYY or MM.DDYyyy
name	global name
prog,prg	program
f,func	function
F	integral of f

1.2.2 Unsupported Entry Points

A large number of entries in this database are not officially supported (i.e. their address is not guaranteed by HP to be stable). However, many of these entries can still be used, provided that the entry address is (or has been) *stable* in all ROM versions.

On the HP49G, two address intervals have been pointed out by Jean-Yves Avenard to be stable, so entries found in these intervals will be added to this database.

On the HP48G, no new ROM versions are to be expected, and all entries can be considered *stable*.

The names of unsupported but stable entries will be *enclosed in single parenthesis*, like (CURSOR@).

1.2.3 More Information

This database has been used to create the entries reference in the second edition of *Programming in System RPL* by Eduardo M. Kalinowski and C. Dominik. In this book, the entry list is embedded into a lot more information about SystemRPL and the HP49G, so if you need additional information, check the book. The main reasons to make also the entry database available is that it is a more compact listing, contains information about ML entries as well and lists the addresses of the entry on many different calculators.

2 HP Objects

2.1 Binary Integers

2.1.1 Built-in BINTS 0-127

03FEF	BINT0	0d 0h aka: ZERO, any
03FF9	BINT1	1d 1h aka: ONE, real, MEMERR
04003	BINT2	2d 2h aka: TWO, cmp
0400D	BINT3	3d 3h aka: THREE, str
04017	BINT4	4d 4h aka: FOUR, arry
04021	BINT5	5d 5h aka: FIVE, list
0402B	BINT6	6d 6h aka: SIX, id, idnt
04035	BINT7	7d 7h aka: SEVEN, lam
0403F	BINT8	8d 8h aka: EIGHT, seco
04049	BINT9	9d 9h aka: NINE, symb
04053	BINT10	10d Ah aka: TEN, sym
0405D	BINT11	11d Bh aka: ELEVEN, hxs
04067	BINT12	12d Ch aka: TWELVE, grob
04071	BINT13	13d Dh aka: TAGGED, THIRTEEN
0407B	BINT14	14d Eh aka: EXT, FOURTEEN, unitob
04085	BINT15	15d Fh aka: FIFTEEN, rompointer

0408F	BINT16	16d 10h aka: REALOB, SIXTEEN
04099	BINT17	17d 11h aka: SEVENTEEN, 2REAL, REALREAL
040A3	BINT18	18d 12h aka: EIGHTEEN
040AD	BINT19	19d 13h aka: NINETEEN
040B7	BINT20	20d 14h aka: TWENTY
040C1	BINT21	21d 15h aka: TWENTYONE
040CB	BINT22	22d 16h aka: TWENTYTWO
040D5	BINT23	23d 17h aka: TWENTYTHREE
040DF	BINT24	24d 18h aka: TWENTYFOUR
040E9	BINT25	25d 19h aka: TWENTYFIVE
040F3	BINT26	26d 1Ah aka: REALSYM, TWENTYSIX
040FD	BINT27	27d 1Bh aka: TWENTYSEVEN
04107	BINT28	28d 1Ch aka: TWENTYEIGHT
04111	BINT29	29d 1Dh aka: TWENTYNINE
0411B	BINT30	30d 1Eh aka: REALEXT, THIRTY
04125	BINT31	31d 1Fh aka: THIRTYONE
0412F	BINT32	32d 20h aka: THIRTYTWO
04139	BINT33	33d 21h aka: THIRTYTHREE
04143	BINT34	34d 22h aka: THIRTYFOUR
0414D	BINT35	35d 23h aka: THIRTYFIVE

04157	BINT36	36d 24h aka: TTHIRTYSIX
04161	BINT37	37d 25h aka: THIRTYSEVEN
0416B	BINT38	38d 26h aka: THIRTYEIGHT
04175	BINT39	39d 27h aka: THIRTYNINE
0417F	BINT40	40d 28h aka: FORTY, FOURTY
04189	BINT41	41d 29h aka: FORTYONE
04193	BINT42	42d 2Ah aka: FORTYTWO
0419D	BINT43	43d 2Bh aka: FORTYTHREE
64B12	BINT44	44d 2Ch aka: FORTYFOUR
64B1C	BINT45	45d 2Dh aka: FORTYFIVE
64B26	BINT46	46d 2Eh aka: FORTYSIX
64B30	BINT47	47d 2Fh aka: FORTYSEVEN
64B3A	BINT48	48d 30h aka: FORTYEIGHT
64B44	BINT49	49d 31h aka: FORTYNINE
64B4E	BINT50	50d 32h aka: FIFTY
64B58	BINT51	51d 33h aka: FIFTYONE
64B62	BINT52	52d 34h aka: FIFTYTWO
64B6C	BINT53	53d 35h aka: FIFTYTHREE, STRLIST, THREEFIVE
64B76	BINT54	54d 36h aka: FIFTYFOUR
64B80	BINT55	55d 37h aka: FIFTYFIVE

64B8A	BINT56	56d 38h aka: FIFTYSIX
64B94	BINT57	57d 39h aka: FIFTYSEVEN
64B9E	BINT58	58d 3Ah aka: FIFTYEIGHT
64BA8	BINT59	59d 3Bh aka: FIFTYNINE
64BB2	BINT60	60d 3Ch aka: SIXTY
64BBC	BINT61	61d 3Dh aka: SIXTYONE
64BC6	BINT62	62d 3Eh aka: SIXTYTWO
64BD0	BINT63	63d 3Fh aka: SIXTYTHREE
64BDA	BINT64	64d 40h aka: BINT40h, SIXTYFOUR, YHI
64BE4	BINT65	65d 41h aka: ARRYREAL
64BEE	BINT66	66d 42h aka: FORTTWO
64BF8	BINT67	67d 43h aka: FOURTHREE
64C02	BINT68	68d 44h aka: SIXTYEIGHT
64C0C	BINT69	69d 45h aka: FOURFIVE
64C16	BINT70	70d 46h aka: SEVENTY
64C20	BINT74	74d 4Ah aka: SEVENTYFOUR
64C2A	BINT79	79d 4Fh aka: SEVENTYNINE
64C3A	BINT80	80d 50h aka: EIGHTY
64C3E	BINT81	81d 51h aka: EIGHTYONE, LISTREAL
64C48	BINT82	82d 52h aka: LISTCMP

64C52	BINT83	83d 53h aka: FIVETHREE
64C5C	BINT84	84d 54h aka: FIVEFOUR
64C66	BINT85	85d 55h aka: 2LIST
64C70	BINT86	86d 56h aka: FIVESIX
64C7A	BINT87	87d 57h aka: LISTLAM
64C84	BINT91	91d 5Bh aka: BINT_91d
64C8E	BINT96	96d 60h aka: BINT_96d
64C98	BINT97	97d 61h aka: IDREAL
64CA2	(BINT98)	98d 62h
64CAC	BINT100	100d 64h aka: ONEHUNDRED
64CB6	(BINT101)	101d 65h
64CC0	BINT111	111d 6Fh aka: char
64CCA	(BINT112)	112d 70h
64CD4	(BINT113)	113d 71h
64CDE	(BINT114)	114d 72h
64CE8	BINT115	115d 73h aka: BINT_115d
64CF2	BINT116	116d 74h aka: BINT_116d
64CFC	(BINT117)	117d 75h
64D06	BINT122	122d 7Ah aka: BINT_122d

2.1.2 Built-in BINTS 127-255

64D10	BINT128	128d 80h aka: BINT80h
64D1A	BINT130	130d 82h aka: BINT130d, BINT_130d, XHI-1
64D24	BINT131	131d 83h aka: BINT_131d, BINT131d, XHI

64D2E	(#8F)	143d 8Fh
64D38	SYMBREAL	145d 91h
64D42	(SYMBCMP)	146d 92h
64D4C	(SYMBSYM)	154d 9Ah
64D56	SYMBUNIT	158d 9Eh
64D60	(backup)	159d 9Fh
64D6A	SYMOB	160d A0h
64D74	SYMREAL	161d A1h
64D88	(SYMLIST)	165d A5h
64D92	SYMID	166d A6h
64D9C	SYMLAM	167d A7h
64DA6	(SYMSYMB)	169d A9h
64DB0	SYMSYM	170d AAh
64DBA	SYMEXT	174d AEh
1CD69	(BINT_AFh)	175d AFh
64DC4	(HXSREAL)	177d B1h
64DCE	(2HXS)	187d BBh
64DD8	BINTC0h	192d C0h
64DE2	2GROB	204d CCh
64DEC	TAGGEDANY	208d D0h
64DF6	EXTREAL	225d E1h
64E00	EXTSYM	234d EAh
64E0A	2EXT	238d EEh
64E14	ROMPANY	240d F0h
64E1E	BINT253	253d FDh
64E28	BINT255d	255d FFh

2.1.3 Built-in BINTS 256-

64E32	REALOBOB	256d 100h
64E3C	#_102	258d 102h
64E46	(#SyntaxErr)	262d 106h
64E50	(BINT_263d)	263d 107h
64E5A	(REALREALOB)	272d 110h
64E64	3REAL	273d 111h
15D6F	(BINT_117h)	279d 117h
64E6E	(Err#Kill)	291d 123h
64E78	(Err#NoLstStk)	292d 124h

64E82	(#NoRoomForSt)	305d 131h
64E8C	(#132)	306d 132h
64E96	(REALSTRSTR)	307d 133h
64EA0	(#134)	308d 134h
64EAA	(#135)	309d 135h
64EB4	(#136)	310d 136h
64EBE	(#137)	311d 137h
64EC8	(#138)	312d 138h
64ED2	(#139)	313d 139h
64EDC	(#13A)	314d 13Ah
64EE6	(#13B)	315d 13Bh
64EF0	(#13D)	317d 13Dh
64EFA	(Err#Cont)	318d 13Eh
64F04	INTEGER337	337d 151h
64F0E	(CMPOBOB)	512d 200h
64F18	(Err#NoLstArg)	517d 205h
64F22	STRREALREAL	785d 311h
64F2C	(ARRYREALREAL)	1041d 411h
64F36	(ARRYREALCMP)	1042d 412h
64F40	(3ARRY)	1092d 444h
64F4A	(ARRYLSTREAL)	1105d 451h
64F54	(ARRYLSTCMP)	1106d 452h
64F5E	(LISTREALOB)	1296d 510h
64F68	(LISTREALREAL)	1297d 511h
64F72	(LISTLISTOB)	1360d 550h
64F7C	(IDREALOB)	1552d 610h
64F86	(IDLSTOB)	1616d 650h
64F90	(FSTMACROROM#)	1792d 700h
64F9A	(PROGIDREAL)	2145d 861h
64FA4	(PROGIDCMP)	2146d 862h
64FAE	(PROGIDLIST)	2149d 865h
64FB8	(PROGIDEXT)	2158d 86Eh
34301	Attn#	2563d A03h
34301	ATTN#	2563d A03h
64FC2	ATTNERR	2563d A03h
64FCC	(SYMREALREAL)	2577d A11h
64FD6	(SYMREALCMP)	2578d A12h
64FE0	(SYMREALSYM)	2586d A1Ah

64FEA	(SYMCOMPREAL)	2593d A21h
64FF4	(SYMCOMPMP)	2594d A22h
64FFE	(SYMCOMPMP)	2602d A2Ah
65008	(SYMIDREAL)	2657d A61h
65012	(SYMIDCMP)	2658d A62h
6501C	(SYMIDLIST)	2661d A65h
65026	(SYMIDEXT)	2670d A6Eh
65030	(SYMSYMREAL)	2721d AA1h
6503A	(SYMSYMP)	2722d AA2h
65044	(3SYM)	2730d AAAh
6504E	(XFERFAIL)	3078d C06h
65058	(PROTERR)	3079d C07h
65062	(InvalServCmd)	3080d C08h
6506C	Connecting	3082d C0Ah
65076	(Retry)	3083d C0Bh
65080	(#CAlarmErr)	3583d DFFh
6508A	EXTOBOB	3584d E00h
03F8B	TYPEREAL	10547d 2933h
03FDB	(TYPEEREL)	10581d 2955h
03FA9	TYPEIDNT	10568d 2948h
03F95	(TYPECMP)	10615d 2977h
03F9F	(TYPELIST)	10868d 2A74h
03FC7	(TYPERRP)	10902d 2A96h
03FBD	(TYPESYMB)	10936d 2AB8h
03FE5	(TYPEEXT)	10970d 2ADAh
03FB3	(TYPECOL)	11677d 2D9Dh
03FA9	TYPEIDNT	10568d 2948h
03FD1	(TYPELAM)	11885d 2E6Dh
65094	#EXITERR	458752d 70000h
6509E	MINUSONE	1048575d FFFFFh

2.1.4 Pushing Several BINTs

641FC	ZEROZERO	(→ #0 #0)
64209	#ZERO#ONE	(→ #0 #1)
6427A	#ZERO#SEVEN	(→ #0 #7)
63AC4	ONEONE	(→ #1 #1)

aka: ONEDUP

6428A	#ONE#27	(→ #1 #27d)
6429D	#TWO#ONE	(→ #2 #1)
642AF	#TWO#TWO	(→ #2 #2)
642BF	#TWO#FOUR	(→ #2 #4)
642D1	#THREE#FOUR	(→ #3 #4)
642E3	#FIVE#FOUR	(→ #5 #4)
64309	ZEROZEROZERO	(→ #0 #0 #0)
6431D	ZEROZEROONE	(→ #0 #0 #1)
64331	ZEROZEROTWO	(→ #0 #0 #2)
62535	DROPZERO	(ob → #0)
64449	(3DROPZERO)	(ob ob ob → #0)
6254E	2DROP00	(ob ob → #0 #0)
62946	DROPONE	(ob → #1)
63A88	DUPZERO	(ob → ob ob #0)
63A9C	DUPONE	(ob → ob ob #1)
63AD8	DUPTWO	(ob → ob ob #2)
63AB0	SWAPONE	(ob ob' → ob' ob #1)
62E3A	ZEROSWAP	(ob → #0 ob)
63079	ZEROOVER	(ob → ob #0 ob)
6351F	ZEROFALSE	(→ #0 F)
62E67	ONESWAP	(ob → #1 ob)
63533	ONEFALSE	(→ #1 F)

2.1.5 Conversion

18CEA	COERCE	(% → #)
62CE1	COERCEDUP	(% → # #)
62E7B	COERCESWAP	(ob % → # ob)
194F7	COERCE2	(% %' → # #')
18CD7	%ABSCOERCE	(% → #)
193DA	(COERCE{ }2)	({ % } → { # })
4F408	C%>#	(C% → # #')
05A03	HXS>#	(hxs → #)
4F3D1	(2HXS>#)	(hxs hxs → # #)
51532	2HXSLIST?	({ hxs hxs' } → # #') Converts list of two hxs to two bints. Generates "Bad Argument Value" for invalid input.
05A51	CHR>#	(chr → #)

2.1.6 Arithmetic Functions

03DBC	#+	(# #' → ##')
25B0B	(#+OVF)	(# #' → ##') $0 \leq \text{result} \leq \text{FFFF}$
03DEF	#1+	(# → #+1)
03E2D	#2+	(# → #+2)
6256A	#3+	(# → #+3)
6257A	#4+	(# → #+4)
6258A	#5+	(# → #+5)
6259A	#6+	(# → #+6)
625AA	#7+	(# → #+7)
625BA	#8+	(# → #+8)
625CA	#9+	(# → #+9)
625DA	#10+	(# → #+10)
625EA	#12+	(# → #+12)
03DE0	#-	(# #' → #-#')
03E0E	#1-	(# → #-1)
03E4E	#2-	(# → #-2)
625FA	#3-	(# → #-3)
6260A	#4-	(# → #-4)
6261A	#5-	(# → #-5)
6262A	#6-	(# → #-6)
03EC2	**	(# #' → **#')
191B9	**OVF	(# #' → **#') $0 \leq \text{result} \leq \text{FFFF}$
03E6F	#2*	(# → #*2)
62691	#6*	(# → #*6)
62674	#8*	(# → #*8)
6264E	#10*	(# → #*10)
03EF7	#/	(# #' → #r #q)
03E8E	#2/	(# → #/2) Rounded down.
637CC	#1--	(# #' → #-#' +1) aka: #-+1
63808	#1--	(# #' → ##' -1) \$1-- is a typo in EXTABLE. aka: #-+1, \$1--
624FB	##2/	(# #' → (##')/2)
627D5	#+DUP	(# #' → ##' ##')

62DFE	#+SWAP	(ob # #' → #+' ob)
63051	#+OVER	(ob # #' → ob #+' ob)
627F8	#-DUP	(# #' → #-#' #-')
62E12	#-SWAP	(ob # #' → #-#' ob)
63065	#-OVER	(ob # #' → ob #-#' ob)
62809	#1+DUP	(# → #+1 #+1)
62E26	#1+SWAP	(ob # → #+1 ob)
1DABB	#1+ROT	(ob ob' # → ob' #+1 ob)
6281A	#1-DUP	(# → #-1 #-1)
5E4A9	#1-SWAP	(ob # → #-1 ob) aka: pull
62FD9	#1-ROT	(ob ob' # → ob' #-1 ob)
28558	#1-UNROT	(ob ob' # → #-1 ob ob')
62E4E	#1-1SWAP	(# → 1 #-1) Returns the bint ONE and the result.
628EB	DUP#1+	(# → # #+1)
626F7	DUP#2+	(# → # #+2)
6292F	DUP#1-	(# → # #-1)
63704	2DUP#+	(# #' → # #' #+') aka: DUP3PICK#+
637F4	DROP#1-	(# ob → #-1)
62794	SWAP#-	(# #' → #' #-')
62904	SWAP#1+	(meta ob → meta&ob) aka: SWP1+
51843	SWAP#1+SWAP	(# ob → #+1 ob)
637E0	SWAP#1-	(# ob → ob #-1)
51857	SWAP#1-SWAP	(# ob → #-1 ob)
5EAF4	(SWAPDROP#1-)	(ob # → #-1)
637A4	SWAPOVER#-	(# #' → #' #-#')
6372C	OVER#+	(# #' → # #' +#)
6377C	OVER#-	(# #' → # #' -#)
63718	ROT#+	(# ob #' → ob #' +#)
63768	ROT#-	(# ob #' → ob #' -#)
637B8	ROT#1+	(# ob ob' → ob ob' #+1)
5FB76	ROT#1+UNROT	(# ob ob' → #+1 ob ob')
62DCC	ROT#+SWAP	(# ob #' → #' +# ob) aka: ROT+SWAP
63740	3PICK#+	(# ob #' → # ob #' +#)
63754	4PICK#+	(# ob1 ob2 #' → # ob1 ob2 #' +#)
62DE5	4PICK#+SWAP	(# ob1 ob2 #' → # ob1 #' +# ob2) aka: 4PICK+SWAP

624BA	#MIN	(# #' → #')
624C6	#MAX	(# #' → #')
03EB1	#AND	(# #' → #') Bitwise AND.

2.1.7 Tests

03D19	#=	(# #' → flag)
03D4E	#<>	(# #' → flag)
03CE4	#<	(# #' → flag)
03D83	#>	(# #' → flag)
03CC7	#0<>	(# → flag)
03CA6	#0=	(# → flag)
622B6	#1<>	(# → flag)
622A7	#1=	(# → flag)
636C8	#2<>	(# → flag)
6229A	#2=	(# → flag)
62289	#3=	(# → flag)
636B4	#5=	(# → flag)
63673	#<3	(# → flag)
636F0	#>1	(# → flag)
		aka: ONE#>
6289B	2DUP#<	(# #' → # #' flag)
628D1	2DUP#>	(# #' → # #' flag)
63385	ONE_EQ	(# → flag) Uses EQ test.
620EB	OVER#=#	(# #' → # flag)
628B5	2DUP#=#	(# #' → # #' flag)
6364B	OVER#0=#	(# #' → # #' flag)
62266	DUP#0=#	(# → # flag)
6365F	OVER#<	(# #' → # flag)
622C5	DUP#1=#	(# → # flag)
636DC	OVER#>	(# #' → # flag)
622D4	DUP#0<>	(# → # flag)
63687	DUP#<7	(# → # flag) Returns TRUE if the argument is smaller than #7.
6362D	2#0=OR	(# # → flag) Returns TRUE if either argument is zero.

2.2 Real Numbers

2.2.1 Built-in Real Numbers

2A487	%-MAXREAL	-9.99E499
2A42E	%-9	-9
2A419	%-8	-8
2A404	%-7	-7
2A3EF	%-6	-6
2A3DA	%-5	-5
2A3C5	%-4	-4
2A3B0	%-3	-3
2A39B	%-2	-2
2A386	%-1	-1
2A4B1	%-MINREAL	-1E-499
2A2B4	%0	0
2A49C	%MINREAL	1E-499
494B4	%.1	.1
495AA	(%.15)	.15
650BD	%.5	.5
650D2	(%-.5)	-.5
2A2C9	%1	1
2A2DE	%2	2
650A8	%e	e
2A2F3	%3	3
2A443	%PI	π
2A308	%4	4
2A31D	%5	5
2A332	%6	6
514EB	(%2PI)	2π
2A347	%7	7
2A35C	%8	8
2A371	%9	9
650E7	%10	10
1CC03	%11	11
1CC1D	%12	12
1CC37	%13	13
1CC51	%14	14
1CC85	%15	15

1CD3A	%16	16
1CD54	%17	17
1CDF2	%18	18
1CE07	%19	19
1CC6B	%20	20
1CCA4	%21	21
1CCC3	%22	22
1CCE2	%23	23
1CD01	%24	24
1CD20	%25	25
1CD73	%26	26
1CD8D	%27	27
320B1	%80	80
415F1	%100	100
650FC	%180	180
65111	(%200)	200
6513B	(%400)	400
65126	%360	360
22352	(%1200)	1200
22367	(%2400)	2400
2237C	(%4800)	4800
22391	(%9600)	9600
0F02D	(%TICKSday)	707788800
2A472	%MAXREAL	9.99E499
514DC	1REV	(→ 6.28318530718) (→ 360.) (→ 400.)

Returns the angle of a full circle, corresponding to the current angular mode.

2.2.2 Built-in Extended Real Numbers

2A4C6	%%0	0
2A562	%.1	0.1
2B3DD	%.4	0.4
2A57C	%.5	0.5
10E68	cfF	0.555... %%5/9 for C↔F conversion.
2A4E0	%%1	1

10E82	cfC	1	For C↔K conversion.
2A4FA	%%2	2	
2A514	%%3	3	
2A458	(%%PI)	π	
2A62C	PI/180	$\pi/180$	
2A52E	%%4	4	
2A548	%%5	5	
0F688	%%2PI	2π	
2B1FF	%%7	7	
2A596	%%10	10	
2B2DC	%%12	12	
2B300	%%60	60	
10E9C	(%%KZERO)	273.15	
10EB6	(%%RZERO)	459.67	

2.2.3 Stack Manipulation Combined with Reals

5198F	(DROP%0)	(ob → %0)
1CA0D	(DROP%1)	(ob → %1)
54B1E	(DROP%0ABND)	(ob → %0)
54A9C	(DROP%1ABND)	(ob → %1)
56AFB	(4DROP%0)	(1 . . . 4 → %0)
50A3B	(UNROT2DROP%0)	(1 2 3 → 3 %0)

2.2.4 Conversion

2A5C1	%>%	(% → %%)
62E8F	%>%%SWAP	(ob % → %% ob)
2A5B0	%%>%	(%% → %)
2B45C	2%>%%	(% % → %% %%)
2B470	2%%>%	(%% %%' → % %')
18DBF	UNCOERCE	(# → %)
1950B	UNCOERCE2	(# # → % %)
63B96	UNCOERCE%%	(# → %%)
19529	(UNCOERCE-{}2)	({#} → {%})
		({# #} → {% %})
5435D	HXS>%	(hxs → %)
05D2C	C%>%	(C% → %re %im)

1B775	(CK%ACOS)	(% → ACOS%)
2AD21	%ATAN	(% → ATAN%)
2ADAE	%SINH	(% → SINH%)
2ADDA	%COSH	(% → COSH%)
2ADED	%TANH	(% → TANH%)
2AE00	%ASINH	(% → ASINH%)
2AE13	%ACOSH	(% → ACOSH%)
1B86C	(CK%ACOSH)	(% → ACOSH%)
2AE26	%ATANH	(% → ATANH%)
1B8DE	(CK%ATANH)	(% → ATANH%)
2A930	%MANTISSA	(% → %mant)
2AE39	%EXPONENT	(% → %expn)
2AF4D	%FP	(% → %frac)
2AF60	%IP	(% → %int)
2AF86	%FLOOR	(% → %maxint <=%)
2AF73	%CEIL	(% → %minint >=%)
2ABDC	%MOD	(% %' → %rem)
2AFAC	(%INT)	(% %' → %rem)
1B30D	(%ARG)	(% %' → %rem)
2AD38	%ANGLE	(%x %y → %ang)
2AD5B	%>%ANGLE	(%x %y → %%ang)
2B529	RNDXY	(% %places → %')
2B53D	TRCXY	(% %places → %')
2AE62	%COMB	(% %' → COMB(%,%'))
2AE75	%PERM	(% %' → PERM(%,%'))
2AE4C	%NFACT	(% → %!)
2B0C4	%FACT	Calculates factorial of number. (% → gamma(%+1))
2AA81	%NROOT	Calculates gamma(x+1). (% %n → %') Calculates the %nth root of the real number. Equivalent to user function XROOT.
2A70E	%MIN	(% %' → %lesser)
2A6F5	%MAX	(% %' → %greater)
62D81	%MAXorder	(% %' → %max %min)
51AB7	(%MAXIMIZE)	(% %' → %max %min)
2AFC2	%RAN	(→ %random) Returns next random number.

2B044	%RANDOMIZE	(%seed →) System level RDZ: seeds the random number generator.
2B07B	DORANDOMIZE	(% →) Stores given number as random number seed.
2A9C9	%OF	(% %' → %' / % * 100)
2AA0B	%T	(% %' → %pcttotal)
2AA30	%CH	(% %' → %pcchange)
2A622	%D>R	(%deg → %rad)
2A655	%R>D	(%rad → %deg)
2B48E	%REC>%POL	(%r %ang → %x %y)
2B4BB	%POL>%REC	(%x %y → %r %ang)
2B4F2	%SPH>%REC	(%r %ang %ph → %x %y %z)
51A71	(2%>%SQR)	(%r %ang %ph → %x %y %z) Does <REF>2%>% and then <REF>%SQR

2.2.6 Extended Real Functions

2A943	%%+	(%% %%' → %%+%%')
2A94F	%%-	(%% %%' → %%-%%')
2A99A	%%*	(%% %%' → %%*%%')
62FED	%%*ROT	(ob ob' %% %%' → ob' %%+%%' ob)
62EA3	%%*SWAP	(ob %% %%' → %%+%%' ob)
63C18	%%*UNROT	(ob ob' %% %%' → %%+%%' ob ob')
2A9E8	%%/	(%% %%' → %%/%%')
63BBE	SWAP%%/	(%% %%' → %%')
63B82	%%/>%	(%% %%' → %)
2AA5F	%%^	(%% %%' → %%^%%')
2AC27	%%SINRAD	(%% → %%')
2AD7C	%%ANGLERAD	(%% → %%')
51A94	(%%SQR)	(%% → %%')
520B2	(2DUP%%R)	
2A8F0	%%ABS	(%% → %%abs)
2AD08	%%ACOSRAD	(%% → %%rad)
2AD4F	%%ANGLE	(%%x %%y → %%ang)
2AD6C	%%ANGLEDEG	(%%x %%y → %%deg)
2ACD8	%%ASINRAD	(%% → %%rad)
2A910	%%CHS	(%% → -%%)
2AA92	%%1/	(%% → 1/%%)

2AC57	%%COS	(%% → %%cos)
2AC68	%%COSDEG	(%%deg → %%cos)
2ADC7	%%COSH	(%% → %%cosh)
2AC78	%%COSRAD	(%%rad → %%cos)
2AB1C	%%EXP	(%% → e^%%)
2AB5B	%%LN	(%% → ln %%)
2AF99	%%FLOOR	(%% → %%maxint) aka: %%INT
2AB94	%%LNP1	(%% → %%ln(%%+1))
2A6DC	%%MAX	(%% %%' → %%max)
2B498	%%R>P	(%%x %%y → %%radius %%angle)
2B4C5	%%P>R	(%%r %%ang → %%x %%y)
2AC06	%%SIN	(%% → %%sin)
2AC17	%%SINDEG	(%%deg → %%sin)
2AD95	%%SINH	(%% → %%sinh)
2AAEA	%%SQRT	(%% → \sqrt{a} %%)
2ACA8	%%TANRAD	(%%rad → %%tan)

2.2.7 Tests

2A8C1	%=	(%%' → flag)
2A8CC	%<>	(%%' → flag)
2A871	%<	(%%' → flag)
2A8B6	%<=	(%%' → flag)
2A88A	%>	(%%' → flag)
2A8A0	%>=	(%%' → flag)
2A76B	%0=	(% → flag)
63BAA	DUP%0=	(% → flag)
2A7CF	%0<>	(% → flag) Can be used to change a user flag into a system flag.
2A738	%0<	(% → flag)
2A799	%0>	(% → flag)
2A7F7	%0>=	(% → flag)
2A81F	%%<	(%% %%' → flag)
2A8AB	%%<=	(%% %%' → falg)
2A87F	%%>	(%% %%' → flag)
2A895	%%>=	(%% %%' → flag)
2A75A	%%0=	(%% → flag)

2A7BB	<code>%%0<></code>	<code>(%% → flag)</code>
2A727	<code>%%0<</code>	<code>(%% → flag)</code>
2A80B	<code>%%0<=</code>	<code>(%% → flag)</code>
2A788	<code>%%0></code>	<code>(%% → flag)</code>
2A7E3	<code>%%0>=</code>	<code>(%% → flag)</code>

2.3 Complex Numbers

2.3.1 Built-in Complex Numbers

524AF	<code>C%0</code>	<code>(0,0)</code>
524F7	<code>C%1</code>	<code>(1,0)</code>
5196A	<code>C%-1</code>	<code>(-1,0)</code>
5193B	<code>C%%1</code>	<code>(%%1,%%0)</code>
5267F	<code>(C%i)</code>	<code>(0,1)</code>
526AE	<code>(C%-i)</code>	<code>(0,-1)</code>

2.3.2 Conversion

519F8	<code>C%>C%</code>	<code>(C% → C%)</code>
05C27	<code>%>C%</code>	<code>(%re %im → C%)</code>
632A9	<code>SWAP%>C%</code>	<code>(%im %re → C%)</code>
51A37	<code>Re>C%</code>	<code>(%re → C%)</code>
519A3	<code>C>Re%</code>	<code>(C% → %re)</code>
519B7	<code>C>Im%</code>	<code>(C% → %im)</code>
51A07	<code>%>C%</code>	<code>(%%re %%im → C%)</code>
519CB	<code>C%>%%</code>	<code>(C% → %%re %%im)</code>
519DF	<code>C%>%%SWAP</code>	<code>(C% → %%im %%re)</code>
51C6B	<code>(SWAP2C%>%)</code>	<code>(C% C%' → %re %im %re' %im')</code>
05DBC	<code>C%>%%</code>	<code>(C% → %%re %%im)</code>

2.3.3 Functions

51A4A	<code>(C%*i)</code>	<code>(C% → C%')</code>
51A5F	<code>(C/i)</code>	<code>(C% → C%')</code>
51C16	<code>(C%C+C)</code>	<code>(C% C%' → C%')</code>
51CFC	<code>(C%C-C)</code>	<code>(C% C%' → C%')</code>

1B48F	(C%C*C)	(C% \rightarrow C%^2)
51EC8	(C%C/C)	(C% C%' \rightarrow C%')
51BD0	(C%C+R)	(C% % \rightarrow C%')
51BF8	(C%R+C)	(% C% \rightarrow C%')
51CE8	(C%C-R)	(C% % \rightarrow C%')
51CD4	(C%R-C)	(% C% \rightarrow C%')
51D4C	(C%C*R)	(C% % \rightarrow C%')
51D60	(C%R*C)	(% C% \rightarrow C%')
51E19	(C%R/C)	(% C% \rightarrow C%')
51E64	(C%C/R)	(C% % \rightarrow C%')
52374	C%C^C	(C% C%' \rightarrow C%')
52360	C%C^R	(C% % \rightarrow C%')
52342	C%R^C	(% C% \rightarrow C%')
52062	C%ABS	(C% \rightarrow %)
51B70	C%CHS	(C% \rightarrow -C%)
51EFA	C%1/	(C% \rightarrow 1/C%)
52107	C%SQRT	(C% \rightarrow $\sqrt{a}C%$)
1B48F	(C%C*C)	(C% \rightarrow C%^2)
520CB	C%SGN	(C% \rightarrow C%/C%ABS)
51BB2	C%CONJ	(C% \rightarrow C%')
52099	C%ARG	(C% \rightarrow %)
52193	C%EXP	(C% \rightarrow e^C%)
521E3	C%LN	(C% \rightarrow ln C%)
522BF	C%LOG	(C% \rightarrow log C%)
52305	C%ALOG	(C% \rightarrow 10^C%)
52530	C%SIN	(C% \rightarrow sin C%)
52571	C%COS	(C% \rightarrow cos C%)
525B7	C%TAN	(C% \rightarrow tan C%)
52804	C%ASIN	(C% \rightarrow asin C%)
52863	C%ACOS	(C% \rightarrow acos C%)
52675	C%ATAN	(C% \rightarrow atan C%)
5262F	C%SINH	(C% \rightarrow sinh C%)
52648	C%COSH	(C% \rightarrow cosh C%)
5265C	C%TANH	(C% \rightarrow tanh C%)
5281D	C%ASINH	(C% \rightarrow asinh C%)
52836	C%ACOSH	(C% \rightarrow acosh C%)
527EB	C%ATANH	(C% \rightarrow atanh C%)
05C72	(%%>C%)	(%%re %%im \rightarrow C%)

51C84	(SWAP2C%>%%)	(C% C%' → %re %im %re' %im')
51C3E	(C%+C)	(C% C%' → C%')
51D10	(C%-C)	(C% C%' → C%')
51DE2	(C%*C)	(C% C%' → C%')
51F13	(C%/C)	(C% C%' → C%')
51C9D	(C%+R)	(C% % → C%')
51CB1	(C%R+C)	(% C% → C%')
51D24	(C%R-C)	(% C% → C%')
51D38	(C%R-C)	(C% % → C%')
51DAB	(C%R+C)	(C% % → C%')
51DBF	(C%R*C)	(% C% → C%')
51F3B	(C%R/C)	(% C% → C%')
51F7C	(C%R/R)	(C% % → C%')
52080	(C%ABS)	(C% → %)
51B91	C%CHS	(C% → -C%)
51BC1	C%CONJ	(C% → C%')

2.3.4 Tests

51B43	C%0=	(C% → flag)
51B2A	C%%0=	(C%% → flag)

2.3.5 Real and Complex Number Functions

35B47	(SWITCHFLOATS)	(B% → ?) Dispatches action based on type. The order is %, C%, %, C%. For example, to change the sign of any float: :: SWITCHFLOATS %CHS C%CHS %CHS C%CHS ;
35B88	(SWITCH2FLOATS)	(L% L% → ?) Works similarly to the above function. The order is % %%, C% %%, % C%, C% C%.
37D19	(F%>L%)	(% → %%) (C% → C%%) Converts float to long float.
37BE9	(L%+)	(L% L% → L%) Adds long real or complex numbers.
37C0C	(L%-)	(L% L% → L%) Subtracts long real or complex numbers.
37C2F	(L%*)	(L% L% → L%) Multiplies long real or complex numbers.

37C52	(L%/)	(L% L% → L%) Divides long real or complex numbers.
37CD3	(B%NEG)	(B% → B%') Changes sign of any number.
37C75	(B%ABS)	(B% → B%') Absolute value of any number.
37DF6	(B%0=)	(B% → flag) Compares any number to zero.

2.4 Character Strings

2.4.1 Built-in Characters

6541E	CHR_00	'\00', CHR 0d 00h The NULL character.
6566A	CHR_Newline	'\0a', CHR 10d 0Ah
65425	CHR_...	'...', CHR 31d 1Fh
65686	CHR_Space	' ', CHR 32d 20h The space character.
6542C	CHR_DblQuote	'"', CHR 34d 22h
65433	CHR_#	'#', CHR 35d 23h
65663	CHR_LeftPar	'(', CHR 40d 28h
65678	CHR_RightPar	')', CHR 41d 29h
6543A	CHR_*	'*', CHR 42d 2Ah
65441	CHR_+	'+', CHR 43d 2Bh
65448	CHR_,	',', CHR 44d 2Ch
6544F	CHR_-	'-', CHR 45d 2Dh
65456	CHR_.	'.', CHR 46d 2Eh
6545D	CHR_/	'/', CHR 47d 2Fh
65464	CHR_0	'0', CHR 48d 30h
6546B	CHR_1	'1', CHR 49d 31h
65472	CHR_2	'2', CHR 50d 32h
65479	CHR_3	'3', CHR 51d 33h
65480	CHR_4	'4', CHR 52d 34h
65487	CHR_5	'5', CHR 53d 35h
6548E	CHR_6	'6', CHR 54d 36h
65495	CHR_7	'7', CHR 55d 37h
6549C	CHR_8	'8', CHR 56d 38h
654A3	CHR_9	'9', CHR 57d 39h
654AA	CHR_:	':', CHR 58d 3Ah

654B1	CHR_;	';', CHR 59d 3Bh
654B8	CHR_<	'<', CHR 60d 3Ch
654BF	CHR_=	'=', CHR 61d 3Dh
654C6	CHR_>	'>', CHR 62d 3Eh
654CD	CHR_A	'A', CHR 65d 41h
654D4	CHR_B	'B', CHR 66d 42h
654DB	CHR_C	'C', CHR 67d 43h
654E2	CHR_D	'D', CHR 68d 44h
654E9	CHR_E	'E', CHR 69d 45h
654F0	CHR_F	'F', CHR 70d 46h
654F7	CHR_G	'G', CHR 71d 47h
654FE	CHR_H	'H', CHR 72d 48h
65505	CHR_I	'I', CHR 73d 49h
6550C	CHR_J	'J', CHR 74d 4Ah
65513	CHR_K	'K', CHR 75d 4Bh
6551A	CHR_L	'L', CHR 76d 4Ch
65521	CHR_M	'M', CHR 77d 4Dh
65528	CHR_N	'N', CHR 78d 4Eh
6552F	CHR_O	'O', CHR 79d 4Fh
65536	CHR_P	'P', CHR 80d 50h
6553D	CHR_Q	'Q', CHR 81d 51h
65544	CHR_R	'R', CHR 82d 52h
6554B	CHR_S	'S', CHR 83d 53h
65552	CHR_T	'T', CHR 84d 54h
65559	CHR_U	'U', CHR 85d 55h
65560	CHR_V	'V', CHR 86d 56h
65567	CHR_W	'W', CHR 87d 57h
6556E	CHR_X	'X', CHR 88d 58h
65575	CHR_Y	'Y', CHR 89d 59h
6557C	CHR_Z	'Z', CHR 90d 5Ah
65694	CHR_['	'[' , CHR 91d 5Bh
6569B	CHR_]'	']' , CHR 93d 5Dh
6568D	CHR_UndScore	'_' , CHR 95d 5Fh
65583	CHR_a	'a', CHR 97d 61h
6558A	CHR_b	'b', CHR 98d 62h
65591	CHR_c	'c', CHR 99d 63h
65598	CHR_d	'd', CHR 100d 64h
6559F	CHR_e	'e', CHR 101d 65h

655A6	CHR_f	'f', CHR 102d 66h
655AD	CHR_g	'g', CHR 103d 67h
655B4	CHR_h	'h', CHR 104d 68h
655BB	CHR_i	'i', CHR 105d 69h
655C2	CHR_j	'j', CHR 106d 6Ah
655C9	CHR_k	'k', CHR 107d 6Bh
655D0	CHR_l	'l', CHR 108d 6Ch
655D7	CHR_m	'm', CHR 109d 6Dh
655DE	CHR_n	'n', CHR 110d 6Eh
655E5	CHR_o	'o', CHR 111d 6Fh
655EC	CHR_p	'p', CHR 112d 70h
655F3	CHR_q	'q', CHR 113d 71h
655FA	CHR_r	'r', CHR 114d 72h
65601	CHR_s	's', CHR 115d 73h
65608	CHR_t	't', CHR 116d 74h
6560F	CHR_u	'u', CHR 117d 75h
65616	CHR_v	'v', CHR 118d 76h
6561D	CHR_w	'w', CHR 119d 77h
65624	CHR_x	'x', CHR 120d 78h
6562B	CHR_y	'y', CHR 121d 79h
65632	CHR_z	'z', CHR 122d 7Ah
656A2	CHR_{	'{', CHR 123d 7Bh
656A9	CHR_}	'}', CHR 125d 7Dh
6564E	CHR_Angle	'∠', CHR 128d 80h
6565C	CHR_Integral	'∫', CHR 132d 84h
65655	CHR_Deriv	'∂', CHR 136d 88h
65639	CHR_→	'→', CHR 141d 8Dh
65640	CHR_<<	'<<', CHR 171d ABh
65647	CHR_>>	'>>', CHR 187d BBh
65671	CHR_Pi	'π', CHR 135d 87h
6567F	CHR_Sigma	'Σ', CHR 133d 85h
656B0	CHR_<=	'≤', CHR 137d 89h
656B7	CHR_>=	'≥', CHR 138d 8Ah
656BE	CHR_<>	'≠', CHR 139d 8Bh

2.4.2 Built-in Strings

055DF	NULL\$	"" Empty string.
65254	SPACE\$	" " aka: tok_
65212	(14SPACES\$)	" " String of 14 spaces.
65238	NEWLINE\$	"\0a" Newline.
2E4F0	CRLF\$	"\0d\0a" Carriage return and line feed.
652B4	(toklparen)	"("
652C0	(tokrparen)	")"
6516A	(tok[)	"["
65150	(tok])	"]"
65176	tok{	"{"
65182	(tok})	"}"
651D6	tok<<	"<<"
651CA	(tok>>)	">>"
6573B	\$_LRParens	"()"
65711	\$_[]	"[]"
65703	\$_{}	"{}"
656F5	\$_<<>>	"<<>>"
6571F	\$_''	''" Two single quotes.
6572D	\$_::	::"
65749	\$_2DQ	""" Two double quotes.
414BD	(\$_:)	": " ": "
65290	tok,	","
65284	tok'	''" One single quote.
652FC	tok-	"-"
6529C	tok.	"."
65308	tok=	"="
25446	tok->	"->"
0FA69	tok_g	"g"
0FA8E	tok_m	"m"
0FACE	tok_s	"s"
6534C	tok0	"0"
65358	tok1	"1"

65364	(tok2)	"2"
65370	(tok3)	"3"
6537C	(tok4)	"4"
65388	(tok5)	"5"
65394	(tok6)	"6"
653A0	(tok7)	"7"
652A8	(tok;)	";"
653AC	tok8	"8"
653B8	tok9	"9"
651BE	tokESC	"\1B" Escape character.
651E2	tokexponent	"E"
65278	tokquote	"" One double quote.
6518E	toksharp	"#"
651A6	(tok\$)	"\$"
651B2	(tok&)	"&"
652D8	(tok*)	"*"
652F0	(tok+)	"+"
652E4	(tok/)	"/"
651EE	(tokanglesign)	"∠"
65320	(tokDER)	"∂"
65244	(\$DER)	"der"
651FA	(tokSIGMA)	"Σ"
65314	(tokSQRT)	"√ <i>a</i> "
6519A	(tokscore)	"_"
65206	(tokWHERE)	" "
652CC	(tok^)	"^"
65797	\$_RAD	"RAD"
657A7	\$_GRAD	"GRAD"
656E5	\$_XYZ	"XYZ"
656D5	\$_R<Z	"R∠Z" "R<angle>Z"
656C5	\$_R<<	"R∠∠" "R<angle><angle>"
65769	\$_EXIT	"EXIT"
65757	\$_ECHO	"ECHO"
6577B	\$_Undefined	"Undefined"
6532C	(tokCTGROB)	"GROB"

6533E	(tokCTSTR)	"C\$"
65260	(tokUNKNOWN)	"UNKNOWN"

2.4.3 Built-in Strings with Stack Manipulation

62D59	NULL\$SWAP	(ob → \$ ob) NULL\$, then SWAP.
04D3E	DROPNULL\$	(ob → NULL\$) DROP then NULL\$.
04D57	(TWO DROPNULL\$)	(ob ob' → NULL\$) 2DROP then NULL\$.
1613F	NULL\$TEMP	(→ \$) Creates null string in temporary memory (NULL\$, then <REF>TOTEMPOB).

2.4.4 Conversion

167E4	#>\$	(# → \$) Creates string from the bint (decimal).
167D8	#:>\$	(# → "#: ") Creates string from the bint and appends a colon and a space. Ex: "1: "
162B8	a%>\$	(% → \$) Converts real number into string using current display mode. aka: a%>\$,
05BE9	ID>\$	(id/lam → \$) Converts identifier into string.
140F1	DOCHR	(% → \$) Creates string of the character with the number specified.
540BB	hxs>\$	(hxs → \$) Uses current display mode and wordsize.
54061	HXS>\$	(hxs → \$) Does <REF>hxs>\$ and then appends base character.

2.4.5 Management

05A75	#>CHR	(# → chr) Returns character with the specified ASCII code.
6475C	CHR>\$	(chr → \$* Strings) Converts a character into a string.
05636	LEN\$	(\$ → #length) Returns length in bytes.

1CA26	(LEN\$>%)	(\$ → %) LEN\$ then UNCOERCE.
627BB	DUPLEN\$	(\$ → \$ #) DUP then LEN\$.
05622	OVERLEN\$	(\$ ob → \$ ob #len) OVER then LEN\$.
63191	NEWLINE\$&\$	(\$ → "\$\0a") Appends newline character to string. aka: NEWLINE&\$
2E4DC	APNDCRLF	(\$ → \$') Appends carriage return and line feed to string.
1782E	(2LEN\$#+)	(\$ \$' → \$ \$' #) Returns sum of length of two strings.
127CA	(DROPDUPLEN\$1+)	(\$ ob → \$ #len+1) Does DROP, then DUP, then LEN\$ and finally #1+.
050ED	CAR\$	(\$ → chr) (\$ → "") Returns first character of string as a string, or NULL\$ for null string.
0516C	CDR\$	(\$ → \$') Returns string without first character, or NULL\$ for null string.
645B1	POS\$	(\$ \$find start# → #pos) (\$ \$find start# → #0) Search for \$find in \$search, starting at position #start. Returns position of \$find or 0 if not found. Same entry as POSCHR.
645B1	POSCHR	(\$search chr #start → #pos) (\$search chr #start → #0) Same entry as <REF>POS\$.
15EF6	(ONEPOS\$)	(\$ \$find/chr → #pos) (\$ \$find/chr → #0) <REF>POS\$ with #start = 1.
1CAD7	(XEQPOS\$)	(\$ \$find/chr → %pos) (\$ \$find/chr → %0) <REF>POS\$ with #start = 1 and followed by UNCOERCE.
645BD	POS\$REV	(\$ \$find #limit → #pos) (\$ \$find #limit → #0) Searches backwards from #limit to #1. Same entry as <REF>POSCHRREV.
645BD	POSCHRREV	(\$seach chr #start → #pos) (\$seach chr #start → #0) Same entry as <REF>POS\$REV.
12770	COERCE\$22	(\$ → \$') If the string is longer than 22 characters, truncates it to 21 characters and appends "...".

45676	Blank\$	(#len → \$) Creates a string with the specified number of spaces.
49709	PromptIdUtil	(id ob → \$) Creates string of the form "id: ob".
127A7	SEP\$NL	(\$ → \$' \$') Separates string at the first newline. '\$' is the substring before the first newline; '\$' the substring after the first newline.
05733	SUB\$	(\$ #start #end → \$') Returns substring between specified positions.
1C8BB	(XEQSUB\$)	(\$ %' → \$') Same as <REF>SUB\$ but uses real numbers as arguments.
63245	#1-SUB\$	(\$ #start #end+#1 → \$') Does #1- and then SUB\$.
63259	1_#1-SUB\$	(\$ #end → \$') Returns substring with the first #end characters. aka: 1_#1-SUB
6326D	LAST\$	(\$ #start → \$') Returns substring from the specified start position to the end (inclusive).
63281	#1+LAST\$	(\$ #start-#1 → \$') Returns substring from the specified start position to the end (exclusive).
62D6D	SUB\$\$SWAP	(ob \$ # #' → \$' ob) SUB\$ then SWAP.
30805	SUB\$1#	(\$ #pos → #') Returns bint with ASCII code of character at the specified position.
61C1C	EXPAND	(hxs #nibs → hxs') Appends #nibs zero nibbles to the hxs.
05193	&\$	(\$ \$' → \$+\$') Concatenates two strings.
63F6A	&\$\$SWAP	(ob \$ \$' → \$+\$' ob) &\$ then SWAP.
62376	!append\$	(\$ \$' → \$+\$') Tries &\$, if not enough memory does !!append\$?.
622E5	!insert\$	(\$ \$' → \$'+\$') Does SWAP then <REF>!append\$.
62F2F	!append\$\$SWAP	(ob \$ \$' → \$+\$' ob) !append\$ then SWAP.
62312	!!append\$?	(\$ \$' → \$+\$') Attempts append "in place" if target is in tempob.
623A0	!!append\$	(\$ \$' → \$+\$') Tries appending "in place".

62394	!!insert\$	(\$ \$' → \$'+\$) Tries inserting "in place".
0525B	>H\$	(\$ chr → \$') Prepends character to string
052EE	>T\$	(\$ chr → \$') Appends character to string.
62BB0	APPEND_SPACE	(\$ → \$') Appends space to string.
622EF	SWAP&\$	(\$ \$' → \$'+\$) Concatenates two strings.
0D304	TIMESTR	(%dt %tm → "dy dt tm") Returns string representation of time, using current format. Example: "WED 06/24/98 10:00:45A"
188D2	(NOT\$)	(\$1 \$2 → \$') Logical NOT.
18873	AND\$	(\$1 \$2 → \$') Logical AND. Errors if strings are not the same length.
18887	OR\$	(\$ \$' → \$'') Logical OR. Errors if strings are not the same length.
1889B	XOR\$	(\$ \$' → \$'') Logical XOR. Errors if strings are not the same length.
18961	(!NOT\$)	(\$ \$' → \$'' ???) Logical NOT "in place".
188E6	(!AND\$)	(\$ \$' → \$'' ???) Logical AND. Does not check if strings are the same length.
188F5	(!OR\$)	(\$ \$' → \$'' ???) Logical OR, does not check if strings are the same length.
18904	(!XOR\$)	(\$ \$' → \$'' ???) Logical XOR. Does not check if strings are the same length.
1410F	(DONUM)	(\$ → CHR) Returns number of first character of string.

2.4.6 Parsing Strings

14137	DOSTR>	(\$ → ?) Internal version of <REF>STR→.
-------	--------	--

238A4	palparse	(\$ → ob T) (\$ → \$ #pos \$' F) Tries parsing a string into an object. If successful, returns object and TRUE, otherwise returns position of error, the offending part of the string \$', and FALSE. If the string contains several arguments, the resulting object is a secondary containing these objects.
0BC6F	!*trior	(F → <SKIP>) (T T → <COLA>)
0BCCF	!*triand	(T T →) (F T → F T <SEMI>)
0BD54	tok8cktrior	(\$1 \$1 → :: \$1 <Ob1> ;) (\$1 \$2 → :: \$1 <Ob2> <Rest> ;)
0BD60	tok8trior	(GNT data \$1 \$1 → :: GNT data GetNextToken ;) (GNT data \$1 \$2 → :: \$1 <Ob1> <Rest> ;)
2534A	nultrior	(NULL\$ → :: ;) (\$ → :: \$ <Ob1> <Rest> ;)
26162	GetNextToken	(hxs-mask \$ #start → hxs-mask \$ #next \$token)
25452	getmatchtok	(hxs-mask \$ #loc \$_tok → hxs-mask \$ #next \$match)
40AD9	Parse.1	
40B2E	ParseFail	(ob \$parsed #pos \$' →) Uses DispBadToken to re-edit the parsed string and displays "Syntax Error".
40B56	(DispBadToken)	(ob \$parsed #pos \$' →) Re-edits the parsed string, positions the cursor to the location of the error. Used by ParseFail.

2.4.7 Decompilation

1795A	!DcompWidth	(# →) Sets the width (in characters) of decompiled strings. This width is used to cut the resulting string (for stack display) or to break it into lines (mostly for editing). Note that most decompilation entries reset this value to the stack or editor width. Use <code>stkdecomp\$w</code> and <code>editdecomp\$w</code> to make sure the current width is used and not changed.
17980	DcompWidth@	(→ #) Recalls the width of decompiled strings (in characters).

159FA	(setStdWid)	(→) Sets DcompWidth to the standard value for stack display, either 19 or 30 characters, depending on system flag 72 (stack minifont). -- Flags: -72
159EB	stkdecomp\$w	(ob → \$) Decompiles for stack display using the current DcompWidth to cut the string if it is too long.
15978	1stkdecomp\$w	(ob → \$) Calls setStdWid and decompiles for stack display (cutting the string if necessary).
41422	>Review\$	(id → \$) Makes a string from the variable name and its contents (decompiled with <REF>Decomp1Line), for display with the review key. If the argument is a command, returns its name.
15B31	editdecomp\$w	(ob → \$) Decompiles entire object for editing. It only decompiles the UserRPL components. Some System RPL entries like <REF>TakeOver are simply skipped, others are written as "External". Breaks the resulting strings into lines using the current DcompWidth.
15A0E	EDITDECOMP\$	(ob → \$) Calls setStdEditWid and the decompiles for editing like <REF>editdecomp\$w.
15B13	DECOMP\$	(ob → \$) Calls <REF>setStdWid and decompiles entire object (UserRPL components only). Breaks the string into lines using DcompWidth as width.
14088	DO>STR	(\$ → \$) (ob → \$) Internal version of →STR.
62B5B	palrompdcmp	(romptr → \$ T) Decompiles a rompointer for the UserRPL stack. If it is a named rompointer, returns the name. Otherwise returns "XLIB n m".

2.4.8 String Tests

0556F	NULL\$?	(ob → flag)
63209	DUPNULL\$?	(ob → ob flag)
142A6	(\$<\$?)	(\$ \$' → flag) String comparizon, alphabetically by character numbers.

1420A	(\$>\$?)	(\$ \$' → flag) String comparizon, alphabetically by character numbers.
142E2	(\$<=\$?)	(\$ \$' → flag) String comparizon, alphabetically by character numbers.
142BA	(\$>=\$?)	(\$ \$' → flag) String comparizon, alphabetically by character numbers.
42C3D	CkChr00	(\$ → \$ flag) Returns FALSE if string contains any null characters.

2.5 HEX Strings

2.5.1 Conversion

059CC	#>HXS	(# → hxs) Length will be five.
543F9	%>#	(% → hxs) Converts real number into hxs. Should be called %>HXS.

2.5.2 General Functions

54039	WORDSIZE	(→ #) Returns the current wordsize as a bint.
53CAA	dostws	(# →) Sets the current wordsize.
055D5	NULLHXS	HXS 0 Puts a null hxs in the stack.
05566	(NULLHXS?)	(hxs → flag) Returns TRUE if the input is a null hxs.
5435D	#>%	(hxs → %)
0518A	&HXS	(hxs hxs' → hxs'') Appends hxs" to hxs'.
61C1C	EXPAND	(hxs #nibs → hxs') Appends #nibs zero nibbles to the hxs.
0EDE1	MAKEHXS	(#nibs → hxs) Makes blank hxs of specified size.
3742D	(!MAKEHXS)	(#nibs → hxs) Makes hxs filled with random data.
05616	LENHXS	(hxs → #nibs) Returns length in nibbles.
05815	SUBHXS	(hxs #m #n → hxs') Returns sub hxs string.

53EA0	bit+	(<i>hxs hxs'</i> → <i>hxs''</i>) Adds two <i>hxs</i> .
54330	bit%#+	(% <i>hxs</i> → <i>hxs'</i>) Adds real to <i>hxs</i> , returns <i>hxs</i> .
54349	bit%#+	(<i>hxs</i> % → <i>hxs'</i>) Adds real to <i>hxs</i> , returns <i>hxs</i> .
53EB0	bit-	(<i>hxs hxs'</i> → <i>hxs''</i>) Subtracts <i>hxs2</i> from <i>hxs1</i> .
542FE	bit%#-	(% <i>hxs</i> → <i>hxs'</i>) Subtracts <i>hxs</i> from real, returns <i>hxs</i> .
5431C	bit%#-	(<i>hxs</i> % → <i>hxs'</i>) Subtracts real from <i>hxs</i> , returns <i>hxs</i> .
53ED3	bit*	(<i>hxs hxs'</i> → <i>hxs''</i>) Multiplies two <i>hxs</i> .
542D1	bit%#*	(% <i>hxs</i> → <i>hxs'</i>) Multiplies real by <i>hxs</i> , returns <i>hxs</i> .
542EA	bit%#*	(<i>hxs</i> % → <i>hxs'</i>) Multiplies <i>hxs</i> by real, returns <i>hxs</i> .
53F05	bit/	(<i>hxs hxs'</i> → <i>hxs''</i>) Divides <i>hxs1</i> by <i>hxs2</i> .
5429F	bit%#/	(% <i>hxs</i> → <i>hxs'</i>) Divides real by <i>hxs</i> , returns <i>hxs</i> .
542BD	bit%#/	(<i>hxs</i> % → <i>hxs'</i>) Divides <i>hxs</i> by real, returns <i>hxs</i> .
53D04	bitAND	(<i>hxs hxs'</i> → <i>hxs''</i>) Bitwise AND.
53D15	bitOR	(<i>hxs hxs'</i> → <i>hxs''</i>) Bitwise OR.
53D26	bitXOR	(<i>hxs hxs'</i> → <i>hxs''</i>) Bitwise XOR.
53D4E	bitNOT	(<i>hxs</i> → <i>hxs'</i>) Bitwise NOT.
53E65	bitASR	(<i>hxs</i> → <i>hxs'</i>) Arithmetic shift one bit to the right. The most significant bit (the sign) does not change.
53E0C	bitRL	(<i>hxs</i> → <i>hxs'</i>) Shifts circularly one bit to the left.
53E3B	bitRLB	(<i>hxs</i> → <i>hxs'</i>) Shifts circularly one byte to the left
53DA4	bitRR	(<i>hxs</i> → <i>hxs'</i>) Shifts circularly one bit to the right.
53DE1	bitRRB	(<i>hxs</i> → <i>hxs'</i>) Shifts circularly one byte to the right.
53D5E	bitSL	(<i>hxs</i> → <i>hxs'</i>) Shifts one bit to the left.
53D6E	bitSLB	(<i>hxs</i> → <i>hxs'</i>) Shifts one byte to the left.

53D81	bitSR	(hxs → hxs') Shifts one bit to the right.
53D91	bitSRB	(hxs → hxs') Shifts one byte to the right.
53EC3	(bitNEG)	(hxs → hxs') Changes sign of hxs.

2.5.3 Tests

544D9	HXS==HXS	(hxs hxs' → %flag) == test
544EC	HXS#HXS	(hxs hxs' → %flag) ≠ test
54552	HXS<HXS	(hxs hxs' → %flag) < test
54500	HXS>HXS	(hxs hxs' → %flag) > test
5452C	HXS>=HXS	(hxs hxs' → %flag) ≥ test
5453F	HXS<=HXS	(hxs hxs' → %flag) ≤ test

2.6 Tagged Objects

05E81	>TAG	(ob \$tag → tagged) Tags an object.
225F5	USER\$>TAG	(ob \$tag → tagged) Maximum of 255 characters in string.
22618	%>TAG	(ob % → tagged) Converts real to string using current display mode and tags object.
05F2E	ID>TAG	(ob id/lam → tagged) Tags object with identifier or lam.
05E9F	({}>TAG)	({ id ob } → tagged)
647BB	TAGOBS	(ob \$tag → tagged) (ob.. { \$.. } → tagged...) Tags one or more objects.
05EC9	(TAG>)	(tagged → ob \$tag)
64775	STRIPTAGS	(tagged → ob) Strips all tags from the object.
647A2	STRIPTAGS12	(tagged ob' → ob ob') Strips all tags from the object in level two.

2.7 Arrays

2.7.1 General Functions

03562	ARSIZE	([] → #) Returns number of elements as a bint.
035A9	DIMLIMITS	([] → {#n #m}) Returns list of array dimensions.
0371D	GETATELN	(# [] → ob T) (# [] → F) Gets one element from array.
0C506	rGETATELN	Gets one element from array referenced by rom- pointer.
0C501	(GETEL)	(#i [] → ob T) (#i [] → F) Gets one element from array.
35D35	(MATIDN)	([F%] → [F%]') Creates identity matrix. Errors if input is not a square matrix.
3745E	SWAPROWS	(M % %' → M') SWAP two rows in matrix. Internal version of xRSWP.
37508	(SWAPCOLUMNS)	([] #m #n → []' #m #n)
9358F	(TYPEARRY@)	([] → #) Returns address of the prolog of the array element type.
03685	(ARRAYEL?)	({#n #m} [] → # T) ({#n #m} [] → F) Returns TRUE if array element exists.
03685	(FINDELN)	({} A → # flag) Return index # of element {} in array.
1DBB0	(BANGARRY)	(el # M → M') Puts el at index # of matrix M.
35CAE	MATCON	([%] % → [%]') ([C%] C% → [C%]') Replace all elements of [F%] by F%.
37E0F	METREDIM	([F%] {#n #m} → [F%]') Redimensions matrix. Removes elements or adds ze- ros as necessary.
357A8	MDIMS	([1D] → #m F) ([2D] → #m #n T) If it is a vector, returns number of elements and FALSE. If it is an array (including arrays with only one line), returns dimensions and TRUE.
62F9D	MDIMSDROP	([2D] → #m #n) MDIMS followed by DROP.

63141	OVERARSIZE	([] ob → [] ob #elts) Does OVER then <REF>ARSIZE.
355B8	PULLREAL	([F%] # → [F%] %) Gets real element.
355C8	PULLCMPEL	([C%] # → [C%] C%) Gets complex element.
3558E	(PULLEL)	([F%] #n → [F%] F%) Gets real or complex element.
35602	(PULLEREALEL)	([F%] #n → [F%] %%) Gets real element then converts to long real.
355D8	(PULLLONGEL)	([F%] #n → [F%] L%) Gets element then converts to long.
35628	PUTEL	([F%] % # → [F%] ') ([C%] C% # → [C%] ') Puts element at specified position. Converts to "short" before. Warning: no copy to tempob first.
3566F	PUTREAL	([F%] % # → [F%] ') Puts real element at specified position. Warning: no copy to tempob first.
356F3	PUTCMPEL	([C%] C% # → [C%] ') Puts complex element at specified position. Warning: no copy to tempob first.
36115	(MAT+)	([F%] [F%] ' → [F%] '') Adds two arrays.
36278	(MAT-)	([F%] [F%] ' → [F%] '') Subtracts two arrays.
3644E	(MAT*)	([F%] [F%] ' → [F%] '') Multiplies two arrays.
36AC3	(MAT/)	([F%] [F%] ' → [F%] '') Divides two arrays.
362DC	(MATFLOAT*)	([F%] F% → [F%] ') Multiplies matrix by float.
363DB	(MATFLOAT/)	([F%] F% → [F%] ') Divides matrix by float.
36444	(MATSQ)	([F%] → [F%] ') Squares matrix.
35F30	(MATCONJ)	([F%] → [F%] ') If a complex array, does the conjugate of all elements. If a real array, does nothing.
35DEB	(MATNEG)	([F%] → [F%] ') Changes sign of all elements of array.
36A99	(MATINV)	([F%] → [F%] ') Reciprocal of all elements of array.
37E0F	MATREDIM	([F%] {#n #m} → [F%] ')
3811F	MATTRN	([F%] → [F%] ') Transposes matrix.

35FA3	(DUP%0CON)	([F%] → [F%] [0%]) DUP then creates a matrix of the same size filled with zeros.
36A48	(MATDET)	([F%] → F%) Calculates determinant of matrix. Generates "Invalid Dimension" error for non-square matrices.
369E9	(MATABS)	([F%] → F%) Returns the scalar magnitude of array.
36705	(MATDOT)	([F%] [F%]' → F%) Returns the dot product of two vectors.
36791	(MATCROSS)	([F%] [F%]' → [F%]') Returns the cross product of two vectors. Generates a "Invalid Dimension" error if inputs are not vectors.
365BB	(MATRSD)	([F%] [F%] [F%] → [F%]) Calculates residuals of solutions of a linear system.
368F4	(MATRNRM)	Row norm.
3690D	(MATCNRM)	([F%] → F%) Column norm.
36039	(MATR>C)	([%re] [%im] → [C%]) Creates complex matrix from real and imaginary parts.
360B6	(MATC>R)	([C%] → [%re] [%im]) Explodes complex matrix into real and imaginary parts.
35F8F	(MATRE)	([F%] → [%re]) Returns (real) matrix with real part of complex numbers. Does nothing if the input is a real matrix.
35FEE	(MATIM)	([F%] → [%im]) Returns (real) matrix with imaginary part of complex numbers. Returns an array of zeros if input is a real matrix.
35E2C	(MATRND)	([F%] % → [F%]') RND on all elements of matrix.
35EA9	(MATTRNC)	([F%] % → [F%]') TRNC on all elements of matrix.
35C2C	(DOARRAYPRG1)	(seco [F%] → [F%]') Evaluates seco for each element in array, then builds array again. Argument for seco will be L%.
35C63	(DOARRAYPRG2)	(seco [F%] [F%] → [F%]') Same as above, but seco has two arguments: one from array1 and another from array2. Arrays must be F%. Arguments for seco will be L%.

2.7.2 Conversion

03442	MAKEARRAY	({#n #m} ob → []) Makes array with all elements initialized to ob.
19294	(>ARRAY)	(F%..F% #n [%F] → [F%]') Copies floats into array.
1D054	XEQ>ARRAY	(F%..F% {#n #m} → [F%]) Makes array with specified dimensions and elements. Does checks first. aka: XEQ>ARRAY
1D02C	(XEQ>VECTOR)	(F%..F% %n → [%F]) Creates a vector.
1D0AB	(DOARRAY>)	([] → F%..F% {#n #m}) Explodes array. Only works for arrays of (normal) real and complex numbers.

2.7.3 Statistics

2C22F	STATCLST	(→) Clears ΣDAT.
2C270	(STATRCL)	(→ ob) Recalls ΣDAT.
2C1F3	(STATSTO)	(ob →) Stores ob into ΣDAT.
2C535	STATN	(→ N) Internal NΣ.
2C58A	STATSMIN	(→ %) Internal MINΣ.
2C558	STATSMAX	(→ %) Internal MAXΣ.
2C571	STATMEAN	(→ %) (→ []) Internal MEAN.
2C5A3	STATSTDEV	(→ %) (→ []) Internal SDEV.
2C5BC	STATTOT	(→ %) (→ []) Internal TOT.
2C5D5	STATVAR	(→ %) (→ []) Internal VAR.
2C675	(STATCOL)	(% %' →) Internal COLΣ.
2C6B6	(STATXCOL)	(n →) Internal XCOL.

2C6CF	(STATYCOL)	(n →) Internal YCOL.
2C6F2	(STATGETXCOL)	
2C706	(STATGETYCOL)	
2C8E6	(STATCOV)	(→ %) Internal COV.
2C940	(STATX)	(→ %) Internal ΣX .
2C959	(STATY)	(→ %) Internal ΣY .
2C972	(STATXX)	(→ %) Internal ΣX^2 .
2C99A	(STATYY)	(→ %) Internal ΣY^2 .
2C9C2	(STATXY)	(→ %) Internal ΣXY .
2CA0D	(STATLR)	
2CB4D	(STATPREDX)	(% → %') Internal PREDX.
2CADA	(STATPREDY)	(% → %') Internal PREDY.
2CCD3	(ColumnMIN)	
2CCBA	(ColumnMAX)	
2CCEE	(ColumnMEAN)	
2CD09	(ColumnTDEV)	
2CCDF	(ColumnTOT)	
2CCFD	(ColumnVAR)	
2C83C	(STATCORR)	(→ %) Internal CORR.

2.8 Unit Objects

2.8.1 Creating Units

10B5E	um*	* marker
10B68	um/	/ marker
10B72	um^	^ marker
10B7C	umP	Char prefix operator
10B86	umEND	Unit end operator
05481	EXTN	(ob1..obn #n → u) Builds a unit object.

2.8.2 General Functions

0FE44	U>NCQ	(u \rightarrow n%% cf%% qhxs) Returns the number, conversion factor to base units and a vector in the form: [kg m A s K cd mol r sr ?] where each element represents the exponent of that unit. For example, 1_N U>NCQ would return: %%1 %%1 [1 1 0 -2 0 0 0 0 0] since it is equivalent to 1.kg*m/s^2
0F33A	UM>U	(% u \rightarrow u') Replaces number part of unit.
0F371	UMCONV	(u1 u2 \rightarrow u1') Change units of unit1 to units of unit2.
0F945	UMSI	(u \rightarrow u') Equivalent to user word UBASE.
0F34E	UMU>	(u \rightarrow % u') Returns number and normalized part of unit.
0F218	UNIT>\$	(u \rightarrow \$) Converts unit to string.
197C8	(UMFACT)	(u1 u2 \rightarrow u) Equivalent to user word UFACT.
10047	U>nbr	(u \rightarrow %) Returns number part of unit.
10065	Unbr>U	(u % \rightarrow u') Replaces number part of unit.
0F41B	TempConv	??? Used by UMCONV for the conversion of temperature units.
1553B	KeepUnit	(% ob ob' \rightarrow % ob) (% ob u \rightarrow u' ob) If the level one object is a unit object, replaces the numeric part of it with the number on level 3. If not, just DROP.

2.8.3 Arithmetic Functions

0F6A2	UM+	(u u' \rightarrow u'')
0F774	UM-	(u u' \rightarrow u'')
0F792	UM*	(u u' \rightarrow u'')
0F823	UM/	(u u' \rightarrow u'')
0FBAB	UM%	(u %percent \rightarrow u')
0FC3C	UM%CH	(u u' \rightarrow %)

0FCCD	UM%T	(u u' → %)
0FB8D	UMMIN	(u u' → u?)
0FB6F	UMMAX	(u u' → u?)
0F8FA	UMXROOT	(u u' → u'')
0F5FC	UMABS	(u → u')
0F615	UMCHS	(u → u')
0F841	(UMINV)	(u → u')
0F913	UMSQ	(u → u')
0F92C	UMSQRT	(u → u')
0FD4A	(UMOPER:)	(u → u')

Evaluates next object with numeric unit part, then builds unit again. For example:

```
:: UOPER: %1/ ;
```

0FCE6	UMSIGN	(u → %)
0FCFA	UMIP	(u → u')
0FD0E	UMFP	(u → u')
0FD22	UMFLOOR	(u → u')
0FD36	UMCEIL	(u → u')
0FD68	UMRND	(u → u')
0FD8B	UMTRC	(u → u')
0F660	UMCOS	(u → u')
0F62E	UMSIN	(u → u')
0F674	UMTAN	(u → u')

2.8.4 Tests

0F584	UM=?	(u u' → %flag)
0F598	UM#?	(u u' → %flag)
0F5AC	UM<?	(u u' → %flag)
0F5C0	UM>?	(u u' → %flag)
0F5D4	UM<=?	(u u' → %flag)
0F5E8	UM>=?	(u u' → %flag)
0F3E4	puretemp?	([] []' → [] []' flag)

Checks of the two arrays both denote pure temperature units, i.e. if both arrays are equal to [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]

2.9 Composites

2.9.1 General Operations

0521F	&COMP	(comp comp' → comp'') Concatenates two composites.
052FA	>TCOMP	(comp ob → comp+ob) Adds ob to tail (end) of composite.
052C6	>HCOMP	(comp ob → ob+comp) Adds ob to head (beginning) of composite.
1AC93	(SWAP>HCOMP)	(ob comp → ob+comp) Does SWAP then >HCOMP.
05089	CARCOMP	(comp → ob_head) (comp_null → comp_null) Returns first object of the composite, or a null composite if the argument is a null composite.
6317D	?CARCOMP	(comp T → ob) (comp F → comp) If the flag is TRUE, does CARCOMP.
05153	CDRCOMP	(comp → comp-ob_head) (comp_null → comp_null) Returns the composite minus its first object, or a null composite if the argument is a null composite.
0567B	LENCOMP	(comp → #n) Returns length of composite (number of objects).
63231	DUPLCOMP	(comp → comp #n) Does DUP then <REF>LENCOMP.
1CA3A	(LENCOMP>%)	(comp → %n) Returns length of composite as a real number.
055B7	NULLCOMP?	(comp → flag) If the composite is empty, returns TRUE.
6321D	DUPNULLCOMP?	(comp → comp flag) Does DUP then <REF>NULLCOMP?.
056B6	NTHELCOMP	(comp #i → ob T) (comp #i → F) Returns specified element of composite and TRUE, or just FALSE if it could not be found.
62B9C	NTHCOMPDROP	(comp #i → ob) Does <REF>NTHELCOMP then DROP.
62D1D	NTHCOMDDUP	(comp #i → ob ob) Does <REF>NTHCOMPDROP then DUP.
71C3B	rNTHELCOMP	(romptr #i → ob T) (romptr #i → F) Returns specified element of composite referenced by the romptr, and TRUE, or just FALSE if it could not be found.

64426	POSCOMP	<pre>(comp ob pred → #i) (comp ob pred → #0) (eg: pred = ' %<)</pre> <p>Evaluates pred for all elements of composite and ob, and returns index of first object for which the pred is TRUE. If no one returned TRUE, returns #0. For example, the program below returns #4:</p> <pre>:: { %1 %2 %3 %-4 %-5 %6 %7 } %0 ' %< POSCOMP ;</pre>
644A3	EQUALPOSCOMP	<pre>(comp ob → #pos) (comp ob → #0)</pre> <p>POSCOMP with EQUAL as test.</p>
644BC	NTHOF	<pre>(ob comp → #i) (ob comp → #0)</pre> <p>Does SWAP then <REF>EQUALPOSCOMP.</p>
6448A	#=POSCOMP	<pre>(comp # → #i) (comp # → #0)</pre> <p>POSCOMP with #= as test.</p>
05821	SUBCOMP	<pre>(comp #m #n → comp')</pre> <p>Returns a sub-composite. Makes all index checks first.</p>
643EF	matchob?	<pre>(ob comp → T) (ob comp → ob F)</pre> <p>Returns TRUE if ob is EQUAL to any element of the composite.</p>
64127	Embedded?	<pre>(ob1 ob2 → flag)</pre> <p>Returns TRUE if ob2 is embedded in, or is the same as, ob1. Otherwise returns FALSE.</p>
644D0	Find1stTrue	<pre>(comp test → ob T) (comp test → F)</pre> <p>Tests every element for test. The first one that returns TRUE is put into the stack along with TRUE. If no object returned TRUE, FALSE is put into the stack. For example, the program below returns %-4 and TRUE.</p> <pre>:: { %1 %2 %2 %-4 %-5 %6 } ' %0< Find1stTrue ;</pre>
644EE	Find1stT.1	<p>Recursive internal function for Find1stTrue.</p>

6452F	Lookup	(ob test comp → nextob T) (ob test comp → ob F) Tests every odd element (1,3,...) in the composite. If a test returns TRUE, the object after the tested one is returned, along with TRUE. If no object tests TRUE, FALSE is returned. For example, the program below returns %6 and TRUE. :: %0 ' %< { %1 %2 %3 %-4 %-5 %6 } Lookup ;
64548	Lookup.1	(ob test → nextob T) (ob test → ob F) Return Stack: (comp →) Lookup with the composite already pushed (with >R) onto the runstream. Called by Lookup.
64593	EQLookup	(ob comp → nextob T) (ob comp → ob F) Lookup with EQ as test.
6480B	NEXTCOMPOB	(comp #ofs → comp #ofs' ob T) (comp #ofs → comp F) Returns object at specified nibble offset from start. If the object is SEMI (i.e., the end of the composite has been reached) returns FALSE. To get the first element, use FIVE as offset value (to skip the prolog). ZERO works as well.

2.9.2 Building

05331	COMPN	(obn..ob1 #n #prolog → comp)
05459	{}N	(obn..ob1 #n → { obn..ob1 })
05445	::N	(ob1..obn #n → :: ob1..obn ;)
0546D	SYMBN	(ob1..obn #n → sym)
54CEF	(SYMBN:)	(ob1..obn #n → symb) Does 'R, SWAP#1+ then SYMBN. Creates a symbolic from the meta in the stack and the next object in the runstream. This object is added to the end of the symbolic.
63F01	top&Cr	(meta1 meta2 → symb) Does top& then <REF>SYMBN: .
5E661	(ONESYMBN)	(ob1..obn #n → symb)
05481	EXTN	(ob1..obn #n → u) Builds a unit object.
5E0DA	P{}N	(ob1..obn #n → {}) Build list with possible garbage collection.

5E111	(P : :N)	(ob1..obn #n → seco) Build seco with possible garbage collection.
5E0A3	(PSYMBN)	(ob1..obn #n → sym) Build symb with possible garbage collection.

2.9.3 Exploding

054AF	INNERCOMP	(comp → obn..ob1 #n)
631E1	DUPINCOMP	(comp → comp obn..ob1 #n)
631F5	SWAPINCOMP	(comp obj → obj obn..ob1 #n)
62B88	INCOMPDROP	(comp → obn..ob1)
62C41	INNERDUP	(comp → obn..ob1 #n #n)
4A95A	ICMPDRPRTDRP	(comp → obn...ob4 ob2 ob1) Does <REF>INCOMPDROP then ROTDROP.
1C973	(XEQLIST>)	(comp → obn..ob1 %n)
636A0	INNER#1=	(comp → obn..ob1 flag)
5E585	INNERTop&	(obn..ob1 #n comp → obm..ob1 #m) Explodes composite and adds to meta: INNERCOMP top& Adds composite objects to meta object.

2.9.4 Lists

055E9	NULL{}	(→ {}) Pushes a null list to the stack.
63A6F	DUPNULL{ }?	({} → {} flag)
23EED	ONE{ }N	(ob → { ob })
631B9	TWO{ }N	(ob1 ob2 → { ob1 ob2 })
631CD	THREE{ }N	(ob1 ob2 ob3 → { ob1 ob2 ob3 })
631A5	#1-{ }N	(ob1..obn #n+1 → {})
1DC00	PUTLIST	(ob #i {} → {}') Replaces object at specified position. Assumes valid #i.
0E461	(INSERTN{ })	({} ob #n → {}') Insert ob at #nth position. Assumes valid #n.
0E4DE	(REMOVEN{ })	({} #n → {}') Removes nth ob. Assumes valid #n.
49CD6	(ROLL{ })	({} → {}') Rolls list elements.
35491	apndvarlst	({} ob → {}') Appends ob to list if not already there.

152FF EqList? (ob →)
 Is ob a list of equations? Returns T if ob is a list of at least two elements, and the second element is not a list itself.

2.9.5 Secondaries

055FD NULL:: (→ :: ;)
 Returns null secondary.

63FE7 Ob>Seco (ob → :: ob ;)
 Does ONE then <REF>::N.

63FCE ?Ob>Seco (ob → :: ob ;)
 If the object is not a secondary, does Ob>Seco.

63FFB 2Ob>Seco (ob1 ob2 → :: ob1 ob2 ;)
 Does TWO then <REF>::N.

632D1 ::NEVAL (ob1..obn #n → ?)
 Does <REF>::N then <REF>EVAL.

5E8DE (argum) (seco → seco #args)
 Returns argument count for secondary. Checks first command, if it is different from CK0, CK1&Dispatch, etc. #5 is returned.

5E9A7 (infarg?) (seco → seco flag)
 Is first command in secondary CKINFARGS?

2.10 Meta Objects

2.10.1 Stack Functions

5E35C (dup) (meta → meta meta)

0326E NDROP (1..n #n →)

63FA6 DROPNDROP (1..n #n ob →)

62F75 #1+NDROP (ob 1..n #n →)
 aka: N+1DROP

169A5 NDROPFALSE (ob1..obn #n → F)

5EB1C psh (meta1 meta2 → meta2 meta1)
 Should be called swap.

5EB58 (rot) (meta1 meta2 meta3 → meta2 meta3 meta1)

5EBDB (unrot) (meta1 meta2 meta3 → meta3 meta1 meta2)

5EBC6 (4roll) (meta1 meta2 meta3 meta4 → meta2 meta3 meta4 meta1)

5EBEA (4unroll) (meta1 meta2 meta3 meta4 → meta4 meta1 meta2 meta3)

5ED45 (5roll) (meta1..meta5 → meta2..meta5 meta1)

5ED5A	(5unroll)	(meta1..meta5 → meta5 meta1..meta4)
5EBFC	(N+1roll)	(META1..METAn+1 #n → META2..METAn+1 META1)
5ED6C	(N+1unroll)	(META1..METAn+1 #n → METAn+1 META1..METAn)
63911	SWAPUnNDROP	(meta1 meta2 → meta2) Should be called swapdrop.
638FD	SWAPUnDROP	(meta1 meta2 → meta2 ob1..obn) Swaps two metas and drops the count. Should be called swapDROP.
5E857	(rotswap)	(meta1 meta2 meta3 → meta2 meta1 meta3)
63F1A	metaROTDUP	(meta1 meta2 meta3 → meta2 meta3 meta1 meta1) Should be called rotdup.
5E870	(4rollunrot)	(meta1 meta2 meta3 meta4 → meta2 meta1 meta3 meta4)

2.10.2 Combining Functions

5E415	top&	(meta1 meta2 → meta1&meta2)
5E4D1	pshtop&	(meta1 meta2 → meta2&meta1)
63F2E	ROUntop&	(meta1 meta2 meta3 → meta2 meta3&meta1)
63F42	roll2top&	(meta1 meta2 meta3 → meta3 meta1&meta2) aka: rolltwotop&
5E3C0	(over&)	(meta1 meta2 meta3 → meta1&meta2 meta3)
5E3AC	psh&	(meta1 meta2 meta3 → meta1&meta3 meta2)
5E843	(overev&)	(meta1 meta2 meta3 → meta2&meta1 meta3)
5E490	(2top&)	(meta1 meta2 meta3 → meta1&meta2&meta3)
5B861	(top&pshtop&)	(meta1 meta2 meta3 → meta2&meta3&meta1)

2.10.3 Meta and Object Operations

62904	SWAP#1+	(meta ob → meta&ob) aka: SWP1+
6119E	DUP#1+PICK	(n..1 #n → n..1 #n n)
5FC24	(pick1)	(ob meta → ob meta ob)
61305	get1	(ob meta → meta ob)
63105	OVER#2+UNROL	(meta ob → ob meta)
5E3E8	(pshm1)	(meta ob → ob #1 meta)
5E401	psh1top&	(meta ob → ob&meta)
5E4A9	pull	(meta&ob → meta ob) aka: #1-SWAP
5EAF4	(pulldrop)	(meta&ob → meta)

5E6BB	(pullpshm1)	(meta&ob → ob #1 meta)
5E4BD	pullrev	(ob&meta → meta ob)
5FA45	(pulldroppull)	(meta&ob1&ob2 → meta ob1)
5CC12	(2pull12DROP)	(meta&ob1&ob2 → meta)
5FA63	(revpulldrop)	(meta&ob1 ob2 → meta ob2)
548AA	(revpull&psh)	(meta&ob1 ob2 → ob1&ob2 meta)
5E706	psh1&	(meta1 meta2 ob → ob&meta1 meta2)
5E7A5	psh1&rev	(meta1 meta2 ob → ob&meta1 meta2)
57432	(addtpsh)	(meta1 meta2 ob → meta1&ob meta2)
10ADB	(UobROT)	(ob meta1 meta2 → meta1 meta2 ob)
10AF9	(unrot1)	(meta1 meta2 ob → ob meta1 meta2)
5E4EA	pullpsh1&	(meta1 meta2&ob → ob&meta1 meta2)
5E503	(pullrev1&)	(meta1 meta2&ob → meta1&ob meta2)
5D6FA	(pshpullpsh1&)	(meta1&ob meta2 → ob&meta2 meta1)
5E67A	pshzer	(meta → #0 meta)
25322	(4psh)	(meta1 ob1..ob4 → ob1&..ob4 meta1)
554B3	(repl%1)	meta2 = ob1&ob2&b3&ob4 (meta&ob → meta&%1)
55607	(repl%-1)	(meta&ob → meta&%-1)
5483C	(COLAkeep1st)	(meta&ob → ob) Returns and (meta&ob ob)
5FC38	(%1pshm1)	(meta → %1 #1 meta)
5E51C	(addt:)	(meta → meta&ob)
5E530	(addt2:)	ob is next ob in runstream. (meta → meta&ob1&ob2)
5E59E	(repl:)	ob is next ob in runstream. (meta&ob → meta&ob')
5E549	(psh1&rev:)	ob is next ob in runstream. (meta1 meta2 → meta1&ob meta2)
5E562	(psh1&rev2:)	ob is next ob in runstream. (meta1 meta2 → meta1&ob1&ob2 meta2)
5DD65	(2psh1&rev:)	ob is next ob in runstream. (meta1 meta2 → meta1&ob meta2&ob)
55477	(replfunc)	ob is next ob in runstream. (meta&ob → meta&LAM1) Uses contents of LAM1.
560ED	xssgneral	(meta1 meta2 → meta1&meta2&LAM1) Uses contents of LAM1.
56101	xnsgeneral	(meta → LAM3&meta&LAM1) Uses contents of LAM1 and LAM3.
5611F	xnsgeneral	(meta → meta&LAM3&LAM1) Uses contents of LAM1 and LAM3.
562BE	(dropaddoper)	(meta1 meta2 meta3 → meta1&meta2&LAM1) Uses contents of LAM1.

56309 (MetaUnCalc) (meta ob → LAM3 #1)
 Uses contents of LAM1 LAM3.

2.10.4 Other Operations

64345 SubMetaOb (meta #start #end → meta')
 Gets a sub-meta. Does range checks.

643BD SubMetaOb1 (ob1..obi..obn #n #i #n #i → ob1..obi #n #i)
)
 This function can be used to take the first i objects of a meta, if you follow it with SWAPDROP. Example:
 :: %1 %2 %3 %4 %5 BINT5
 BINT3 BINT5 BINT3
 SubMetaOb1 ;
 results in:
 %1 %2 %3 #5 #3

5F996 (tailpsh) (meta #n → meta1 meta2)
 Pushes n-1 last objects in meta to meta1.

28296 metatail (ob1..obn-i..obn #i #n+1 → ob1..ob..obn-i #n-i obn-i+1..obn #i)
 #n is the count of the objects in meta. Takes the last #i elements of meta and creates a new one.
 Example:
 :: %1 %2 %3 %4 %5
 BINT2 BINT6 metatail ;
 Results:
 %1 %2 %3 #3 %4 %5 #2

584B2 (MEQU?) (meta1 meta2 → meta1 meta2 flag)
 If the metas are equal (i.e., same count and equal objects) returns TRUE.

5768A (ObInMeta?) (meta ob → meta ob flag)
 Returns TRUE if ob is equal to some ob in meta.

55314 (?addinver:) (meta&Nob → meta)
 (meta → meta&1LAM)
 If next object in the runstream is equal to first object of meta, drops that object. Otherwise, adds 1LAM to meta.

5540E (?addrevert) (meta&1LAM → meta&1LAM)
 (meta → meta&1LAM)
 Adds 1LAM to meta, if not already there.

5613D (?addsimir) (meta meta → meta)

58715 (NoIdsInMeta?) (meta → meta flag)
 If meta has any ids, lams or secondaries starting with CK0, returns FALSE.

5AD08 (dvars?) (meta → meta flag)
 Returns TRUE if meta contains any LAM dvar.

5670F	(>dvars)	(meta1 meta2 → meta1&meta2') All ids in meta2 matching lam 'dvar contents are changed to LAM.'dvar. (meta1 can be #0).
5AC86	(dvars>)	(meta → meta') Lam 'dvars: are changed to 1LAM)
50F60	(dropDROPf)	(ob meta → F)
57419	(DROP2dropf)	(meta1 meta2 ob → F)
57405	(2DROP2dropf)	(meta1 meta2 ob1 ob2 → F)
5551C	(Repl0)	(meta → %0 #0)
55535	(Repl1)	(meta → %1 #1)
5554E	(Repl-1)	(meta → %-1 #1)
56183	(2Repl0)	(meta1 meta2 → %0 #1)
561D8	(2Repl-1)	(meta1 meta2 → %-1 #1)
5643A	(DropRepl0)	(meta ob → %0 #1)
5499F	(Repl0ABND)	(meta → %0)

2.11 Symbolics

2.11.1 General Operations

0546D	SYMBN	(ob1..obn #n → sym)
54CEF	(Cr)	ob1..obn #n -> symb Does 'R, SWAP#1+ then <REF>SYMBN . Creates a symbolic from the meta in the stack and the next object in the runstream. This object is added to the end of the symbolic.
055F3	(NULLSYMB)	(→ sym) Puts a null algebraic in the stack.
5E067	(SINNER)	(sym → meta) (ob → ob #1) If the argument is a symbolic, does INNERCOMP, otherwise ONE. Note that ob #1 is a meta object with only one object.
5E30C	(2SINNER)	(ob1 ob2 → meta1 meta2) SINNER for two objects.
5E2F8	(2SINNERtop&)	(ob1 ob2 → meta) Does <REF>2SINNER then <REF>top& .
5E32A	(SINNERMETA)	(meta → meta') Explodes each object in meta with SINNER and merges the result with top&.
5F2A3	(EXPLODE)	(ob → meta) Uses recursive calls to SINNER to explode object.

5F2EE	(IMplode)	(meta → ob) Builds symbolic obeying VUNS properties (UNSYM element), checking fcnapply, etc. Does not build symbolic if result is a single object valid in symbolics.
5E652	symcomp	(ob → ob') If ob is symbolic, does nothing, otherwise ONE SYMBN.
5A01D	SWAPcompSWAP	(ob ob' → ob'' ob') Does SWAP symcomp SWAP.
5E085	(CKSYMBN)	(meta → ob) If size is not one, does SYMBN, else DROPSYM.
5F384	(DROPSYM)	(ob1 ob2 → ob) Drops ob2, if ob1 if symf does nothing, else does ONESYMBN.
1CF2E	(SPLITEQ)	(sym → arg1 arg2) Internal version of EQ→.
1CFD0	(EXPR>)	(sym → arg1..argn %n ob) Internal version of OBJ→.
1578D	CRUNCH	(ob → %) Internal version of →NUM.
22F68	(SYMCRUNCH1)	(ob → %) If id does <REF>XEQRCL , then does <REF>CRUNCH for all object types.
22F86	(SYMCRUNCH2)	(ob1 ob2 → % ob2) <REF>SYMCRUNCH1 for the object in level two.
353AB	(FINDVAR)	(sym → { }) Returns a list of the variables of the equation, recursing into programs and functions in the equation.
5A036	uncrunch	(→) Clears numeric results flag (system flag 3) for the next command only. Example: SYMCOLCT = :: uncrunch colct ; --
545A0	cknumdsptch1	Flags: -3 (sym → symf) Used by one argument functions to evaluate a symbolic or numeric routine according to numeric results flag. Usage: :: cknumdsptch1 <sym> <num> ; If numeric mode, CRUNCH is applied to the level one object and COLA is applied to <num>. If symbolic mode, ckseval1: is called. Example: :: cknumdsptch1 MetaRE xRE ; -- Flags: -3

54DBC	(ckseval1:)	(symf' → symf') Binds next two objects in the runstream to LAMxSYMfcn and LAMxfnc. Explodes symf, then evaluates next on Meta, then builds ob with CKSYMBN. If symf is equation next is evaluated on both sides, then equation is rebuilt (ckevaleq1).
54E2A	(ckevaleq1)	(meta&= → sym) Evaluates 2LAM on both sides of equation, rebuilds symbolic and abandons temporary environment.
558DC	sscknum2	(sym sym → symf) Used by two argument functions to evaluate function according to current numeric mode. Usage: :: sscknum2 <sym> <num> ; In numeric mode both arguments are CRUNCHED and <num> is COLAd. Else, cksseval2: is called. Example: SYM+ = :: sscknum2 Meta+ x+ ;
558F5	sncknum2	(sym % → symf) Usage: :: sncknum2 <sym> <num> ; In symbolic mode uses cksneval2:. Example: SYM+0 = :: sncknum2 Meta+Con x+ ;
5590E	nscknum2	(% sym → symf) Usage: :: nscknum2 <sym> <num> ; In symbolic mode uses cknseval2:. Example: 0+SYM = :: nscknum2 Con+Meta x+ ;
55657	(cknum2:)	(symf symf → symf) Used by the three above functions to determine (and possibly to CRUNCH) the program to COLA.
557EC	(cksseval2:)	(sym sym → symf) Binds next two objects in the runstream to LAMxSYMfcn and LAMxfnc. Explodes the objects in the stack, and evaluates next object in the runstream. If either is an equation, ckevaleq2 is called. Rebuilds one symbolic.
5576F	(cksneval2:)	(sym % → symf) Binds % and next two objects in the runstream to LAMsc1, LAMxSYMfcn and LAMxfnc. Explodes sym, evaluates LAMxSYMfnc, rebuilds symbolic. If sym is equation, ckevaleq1 is called.
5575B	(cknseval:)	(% sym → symf) Does SWAP then <REF>cknseval2: .

58CE4	(parameval)	(sym param → ?) Ensures sym is symbolic (using symcomp), then executes param on each element of symbolic. param is bound to 1LAM during the loop. param should return a flag. If TRUE, or if the object in level 1 is not an operator the loop continues, else possible COLCT property is executed. (Better return TRUE always).
58CEE	(eval)	(sym → ?) Like <REF>parameval, but without binding of a new param. Use this for recursive evaluation with the same parameter. (See SHOWLS and showparam for examples).
5918A	(evalTRUE)	(sym → ? T) Used for recursive parameval.

2.11.2 Mathematical Operations

55F2B	(SYM+0)
55F44	(0+SYM)
55F5D	(SYM+)
55F76	(SYM-0)
55F85	(0-SYM)
55F8F	(SYM-)
55FC1	(SYM*0)
55FDA	(0*SYM)
55FF3	(SYM*)
5600C	(SYM/0)
56025	(0/SYM)
5603E	(SYM/)
55EE0	(SYM^0)
55EF9	(0^SYM)
55F12	(SYM^)
56057	(SYM%MOD)
56070	(%SYMMOD)
56089	(SYMMOD)
55E95	(SYM%MIN)
55EAE	(%SYMMIN)
55EC7	(SYMMIN)
55E4A	(SYM%MAX)
55E63	(%SYMMAX)

55E7C	(SYMMAX)
55C3D	(SYM%%OF)
55C56	(%SYM%OF)
55C6F	(SYM%OF)
55C88	(SYM%%CH)
55CA1	(%SYM%CH)
55CBA	(SYM%CH)
55CD3	(SYM%%T)
55CEC	(%SYM%T)
55D05	(SYM%T)
55D1E	(SYM%COMB)
55D37	(%SYMCOMB)
55D50	(SYMCOMB)
55D69	(SYM%PERM)
55D82	(%SYMPERM)
55D9B	(SYMPERM)
55DB4	(SYM%RND)
55DCD	(SYMRND)
55DE6	(RNDSYM)
55DFE	(SYM%TRNC)
55E18	(TRCNYM)
55E31	(SYMTRCN)
560A2	(SYM%XROOT)
560BB	(%SYMXROOT)
560D4	(SYMXROOT)
54EEB	(SYMNEG)
54F04	(SYMABS)
54F68	(SYMSIGN)
54F36	(SYMINV)
5518E	(SYMIP)
551A7	(SYMFP)
551C0	(SYMFLOR)
551D9	(SYMCEIL)
5520B	(SYMMANT)
551F2	(SYMEXPONENT)
54AE0	(SYMRE)
54EB9	(SYMIM)
54F1D	(SYMCONJ)

54ED2	(SYMNOT)
54F9A	(SYMSQ)
54F81	(SYMSQRT)
54FB3	(SYMSIN)
54FCC	(SYMCOS)
54FE5	(SYMTAN)
55049	(SYMASIN)
55062	(SYMACOS)
5507B	(SYMATAN)
54FFE	(SYMSINH)
55017	(SYMCOSH)
55030	(SYMTANH)
55094	(SYMASINH)
550AD	(SYMACOSH)
550C6	(SYMATANH)
550F8	(SYMLN)
55143	(SYMLNP1)
550DF	(SYMEXP)
5515C	(SYMEXPM)
55111	(SYMLOG)
5512A	(SYMALOG)
55175	(SYMFACT)
55224	(SYMD>R)
5523D	(SYMR>D)
54F4F	(SYMARG)
55256	(SYMUBASE)
5226F	(SYMUVAL)
5599A	(SYM%AND)
559B3	(%SYMAND)
559CC	(SYMAND)
559E5	(SYM%OR)
559FE	(%SYMOR)
55A17	(SYMOR)
55A30	(SYM%XOR)
55A49	(%SYM XOR)
55A62	(SYM XOR)
55A7B	(SYMFLOAT==)
55AAD	(SYM==)

55A94 (FLOATSYM==)
55AC6 (SYMFLOAT<>)
55ADF (FLOATSYM<>)
55AF8 (SYM<>)
55B11 (SYM%<)
55B2A (%SYM<)
55B43 (SYM<)
55B5C (SYM%>)
55B75 (%SYM>)
55B8E (SYM>)
55BA7 (SYM%<=)
55BC0 (%SYM<=)
55BD9 (SYM<=)
55BF2 (SYM%>=)
55C0B (%SYM>=)
55C24 (SYM>=)
56331 (Con+Meta)
56543 (Meta+Con)
56160 (Meta+)
56566 (Meta-Con)
56359 (Con-Meta)
56174 (Meta-)
56589 (Meta*Con)
56390 (Con*Meta)
561BA (Meta*)
565CF (Meta/Con)
563DB (Con/Meta)
56214 Meta/
5645D (Meta^Con)
562FA (Con^Meta)
5660B (MetamodCon)
5642B (ConmodMeta)
56250 (Metamod)
553D2 (MetaNEG)
555B2 (MetaABS)
553EB (MetaINV)
5542C (MetaRE)
55495 (MetaIM)

55567 (MetaCONJ)
 555E9 (MetaSQ)
 5533C (MetaSIN)
 55378 (MetaCOS)
 553A5 (MetaTAN)
 5529C (MetaSINH)
 552B0 (MetaCOSH)
 552C4 (MetaTANH)
 552D8 (MetaEXP)
 55300 (MetaEXPM)
 552EC (MetaALOG)

2.11.3 Collection

57D90 SYMCOLCT (symf \rightarrow symf)
 :: uncrunch colct ;

57DA4 (colct) (symf \rightarrow symf)
 Basic collection function, does not check numeric re-
 sults flag. Disassembly:
 :: EXPLODE
 pshzer colfac
 pshzer colrev
 ATTNFLAG@ #0<> case
 :: CKSYMBN CKONOLASTWD
 ?ATTNQUIT ;
 pshzer colunfac
 SYMN COLA coleval
 ;
 --
 Flags: -3

587AA (colfac) (meta1 meta2 \rightarrow meta')
 Appends objects in meta2 tail to meta1 tail replacing
 all -, /, NEG, INV and SQ with +, *, ^, and -1 as a
 possible factor. Example rules:
 'SQ(A)' 'A^2'
 '-A' '-1*A'
 'A-B' 'A+-1*b'
 'A/B' 'A*B^-1'

57E08	(colrev)	(meta1 meta2 → meta') Appends objects in meta2 to tail of meta1 collecting numeric factors, ordering terms according to a comparison function, collecting numeric terms to front. Only + and * factors are checked. Sub-routines used by this function:
58511	(MetaLess?)	(meta1 meta2 → meta1 meta2 flag)
58525	(MetaMore?)	(meta1 meta2 → meta1 meta2 flag)
585A7	(BodyMore?)	(ob1 ob2 → flag)
58A20	(colunfac)	(meta1 meta2 → meta') Appends objects in meta2 to head of meta1 converting ^, + and * to / and - when suitable.
58CDA	(colevel)	(ob → ob') Passes FALSE as parameter to parameval. Thus eval uses ?COLCT to check special evaluation.

2.11.4 Expansion

57A0C	(SYMEXPAN)	(symf → symf) Expands symbolic or float
57A48	(expan)	(meta1 meta2 meta3 → meta) Expands meta3. Successful part is added to tail of meta2. Calls expan1 and larg until meta3 becomes empty.
57AA2	(expan1)	(meta → meta1 meta2) Expands meta. Meta1 is the unsuccessful part, meta2 the successful part (could be just and operator). Sub-expanders:
57B63	(?expan^)	If ^ then expands (returns if successful.)
57AB6	(expansq)	Expands SQ.
57B4C	(?expanneginv)	Prevents Meta->() from expanding [Expr INV NEG].
57B01	(?expanapp)	If <REF>xFCNAPPLY then tries calling <REF>?EXPAN .
57C71	(expansum^)	Expands (A+B)^2 or (A-B)^2
57CF8	(NXTPOT%)	(% → flag %' T) (% → % F) Returns next number when expanding ^. The flag indicates wheter %0>. Do not use for %0.

2.11.5 Integration

1F201	(XEQINTEGID)	(ob ob ob id/lam → symf)
1F27A	(XEQINTEG)	(ob ob ob QN → symf)
5AAC7	(SYMINTEG)	(symf symf symf QN → symf)
5662E	(NUMINTEG)	(symf QN symf_lo symf_hi → %)
52C36	(CALCINTEG)	(seco %precision %lo %hi → %integral %error) Low level numeric integration. If %low = %hi returns %0 %0. Checks that $1E-12 \leq \%accuracy \leq 1$. seco gets % as input and should return one value.
5ACC7	(intg)	(#0 #0 meta → meta_ok meta_fail) Integrates meta where variable of integration has been changed to LAMdvar. Meta objects should be merged by addition. Use <code>colunfac</code> to resume /, -, etc. from *, +.
5D0C2	(forceadd)	(meta → meta') Forces top level operators to be +, NEG when possible by changing from -, +, NEG. Attempts to arrange rightmost term to be second argument for top +. Example: 'A+(B+C)' 'A+B+-C'
5B659	(forcemul?aga)	(meta → meta') Recursive Meta<-D, MetaD-> and forcemul calling. If any operation was successful AGAIN is executed.
5B717	(forcemul)	(meta → meta') Forces top level operator to be + and NEG when possible by changing from / and INV. LAMdvar is ordered specially.
5AFAB	(intg1)	(M_ok M_fail M_temp meta → M_ok' M_fail' M_temp) Integrates meta, ok part is adds to meta1 (meta3 is the next part to integrate in the top level loop.)
5B0FA	(intg1ok)	(M1 M2 M3 M4 → M1' M2 M3 T) Adds M4 to M1. (Successful intg1).
5B09B	(intg1fail)	(M1 M2 M3 M4 → M1 M2' M4 T) Adds M4 to M2. (Unsuccessful intg1).
5B0CD	(intgconst)	(M_ok M_fail M_temp meta → M_ok' M_fail' M_temp) Integrates constant to meta. (dvars? gives FALSE).
5B131	(intglinear)	(M1 M2 M3 M4 → M1' M2 M3) Integrates linear term (M4).

5B140	(intgaddlin)	(meta #loc → meta') Adds 2 ² to LAMdvar in meta at stack level #loc.
5AD80	(linear?)	(meta #level → meta' T) (meta #level → meta' #loc F) Is meta linear in LAMdvar? #level is first location of LAMdvar obtained from dvars? :: linear DUP IT SWAPDROP ;
5AD9E	(linear)	(meta #level → meta #loc flag) :: linear SWAPDROP ;
5AD6C	(linear!)	(meta #level → meta' flag)

2.11.6 Derivatives

7DBE2	D/D*	Derivative of multiplication.
7DBED	D/D+	Derivative of addition.
7DBF8	D/D-	Derivative of subtraction.
7DC03	D/D/	Derivative of division.
7DC72	D/D=	Derivative of equality.
7DC7D	D/DABS	Derivative of ABS.
7DCA1	D/DACOS	Derivative of ACOS.
7DCAC	D/DACOSH	Derivative of ACOSH.
7DCB7	D/DALOG	Derivative of ALOG.
7DE1C	D/DAPPLY	
7DCC2	D/DARG	Derivative of ARG.
7DCCD	D/DASIN	Derivative of ASIN.
7DCD8	D/DASINH	Derivative of ASINH.
7DCE3	D/DATAN	Derivative of ATAN.
7DCEE	D/DATANH	Derivative of ATANH.
7DCF9	D/DCHS	Derivative of CHS.
7DD17	D/DCONJ	Derivative of CONJ.

7DD35	D/DCOS	Derivative of COS.
7DD40	D/DCOSH	Derivative of COSH.
7DDF0	D/DDER	Derivative of derivative.
7DD4B	D/DEXP	Derivative of EXP.
7DD82	D/DIFTE	Derivative of IFTE.
7DE06	D/DINTEGRAL	Derivative of integral.
7DD56	D/DINV	Derivative of INV.
7DD61	D/DLN	Derivative of LN.
7DD6C	D/DLNP1	Derivative of LNP1.
7DD77	D/DLOG	Derivative of LOG.
7DD8D	D/DSIN	Derivative of SIN.
7DD98	D/DSINH	Derivative of SINH.
7DDA3	D/DSQ	Derivative of SQ.
7DDAE	D/DSQRT	Derivative of SQRT.
7DE11	D/DSUM	Derivative of SUM.
7DDB9	D/DTAN	Derivative of TAN.
7DDC4	D/DTANH	Derivative of TANH.
7DDFB	D/DWHERE	
7DDCF	D/D [^]	Derivative of power.
7DDDA	D/D [^] X	
7DDE5	D/D [^] Y	

2.11.7 Other Functions

1F38B	(SYMWHERE)	(symf {} → symf)
1F439	(XEQSYMWHERE)	(symf QN1 id1..QNn idn → symf)
1F43E	(CKWHEREARGS)	checks pairs of quoted names Checks pairs of quoted names/ids.

547B5	SYMBWHERE	(symf QN1 id1..QNn idn #2n+1 → symf)
547E2	(WHERE1)	(QN1 id1..QNn idn #n meta1 → symf) Used when meta size is 1.
54887	(WHERE2)	(QN1 id1..QNn idn #2 meta2 → symf)
58D75	SYMSHOW	(sym id/lam → symf)
20B00	XEQSHOWLS	(sym {} → symf)
5910B	(SHOWLS)	(sym {names} → symf) See this for a good example of recursive parameval.
1A4A3	(%IFTE)	(% ob1 ob2 → ?)
54564	(SYMIFTE)	(sym symf symf → symf) Uses cknnumdsptch1 with:
54609	(MetaIFTE)	
54653	(NumIFTE)	
591AD	(SYMQUAD)	(sym id → symf) Avoids the obvious in solving a quadratic equation.
595DD	(SYMTAYLR)	(sym id % → symf) Calculates taylor polynomial.
57293	(SYMISOL)	(sym id → symb) Isolate a variable.
1F113	(XEQSYMDERCON)	(QN %/C%/u → symf)
1F0F5	(XEQSYMDERSTEP)	(QN sym → symf)
54977	(SYMDERSTEP)	(QN sym → symf) No CKSYMBTYPE check.
54954	(SYMDER)	(sym sym → symf)
56949	(SYMSUM)	(sym sym sym ob → symf)
56A06	(SYM%SUM)	(sym sym % ob → symf)
56A4C	(%SYMSUM)	(sym % sym ob → symf)
56AC9	(%%SUM)	(sym % % ob → symf)

2.11.8 Meta Symbolics Functions

5BC94	(addt+)	(meta → meta&+)
5BC67	(addt-)	(meta → meta&-)
5CD16	(addt*)	(meta → meta&*)
5CD2A	(addtNEG)	(meta → meta&NEG)
5CD3E	(addtINV)	(meta → meta&INV)
5BCC1	(repl/)	(meta&ob → meta&/)
5BCEE	(repl*)	(meta&ob → meta&*)
5ACD6	(M1st+?Drp)	(meta&+ → meta)

5BC5D	(meta+)	(meta&NEG → meta&-)
5BC8A	(meta-)	(meta&NEG → meta&+)
5BCB7	(meta*)	(meta&INV → meta&/)
5BCE4	(meta/)	(meta&INV → meta&*)
5BD3E	(drpmeta+)	(meta&NEG&ob → meta&-) (meta&ob → meta&+)
5BD57	(drpmeta-)	(meta&NEG&ob → meta&+) (meta&ob → meta&-)
5BD70	(drpmeta*)	(meta&INV&ob → meta&/) (meta&ob → meta&*)
5BD89	(drpmeta/)	(meta&INV&ob → meta&*) (meta&ob → meta&/)
5BBE5	(metaneg)	(meta&NEG → meta) (meta → meta&NEG)
5BC3F	(metainv)	(meta&INV → meta) (meta → meta&INV)
5BC03	(metaneglft)	(meta → meta') metaneg on left sub-expression.
5BC21	(metainvlft)	(meta → meta') metainv on left sub-expression.
5EA9F	pshzerpsharg	(meta → M_last M_rest) Pushes last sub-expression in meta. If meta is a valid expression M_rest will be empty.
63F92	pZpargSWAPUn	(meta → M_rest M_last) <REF>pshzerpsharg then <REF>psh .
63F56	plDRPpZparg	(meta&ob → M_last M_rest) Drops ob then calls <REF>pshzerpsharg .
5E68E	(pargop)	(meta → M_last&op M_rest) Pushes last sub-expression ignoring first object in meta. Thus op is +, -, etc. and M_last is their second argument.
5EAC2	(larg)	(meta → M_rest M_last) Splits last sub-expression from meta.
5E6F2	(parg&)	(meta1 meta2 → meta1&M_last M_rest)
5CCEE	(larg&)	(meta1 meta2 → meta1&M_rest M_last)
5CBF9	(drppargtop&)	(meta&ob → M_last&M_rest)
57F4B	(swappargunrot)	(meta1 meta2 → M_rest meta2 M_last)
1CF42	(drppargsym)	(meta&ob → 'M_rest' 'MetaLast') Buids objects with PSYMBN. Will give invalid expressions if ob is not a two-argument function.
5F926	(splitup)	(meta #n #m → meta #level) Calculates stack level of last object to be included when splitting last #m sub- expressions from meta starting from stack level n. (2 1 would give level of first object in the last sub-expression.)

5F96E	(splitdown)	(meta #n #m → meta #lowlevel #args+1) Seeks stack level n-1 downwards for extra operators for #m expressions. #lowlevel is the stack level of the extra operator. #args indicates how many expressions the lowlevel operator is still missing.
558BE	(?spliteq)	(meta1&meta2&= → meta2 meta1) If meta contains =, splits two sides, otherwise DUP.
58C02	(count+)	(meta → meta #0)
58C0E	(count*)	(meta &+&+..&+ → meta #n) (meta → meta #0) (meta &*&*..&* → meta #n) Same as count+ for *.
5BE56	(MetaMulInv)	Simplify combinations of INV and * (using /).
58A61	(colinv1)	([expr1 INV expr2 INV *] → [expr1 expr2 /])
58A93	(colinv2)	([expr INV *] → [expr /])
58AAC	(colinv3)	([expr1 INV expr2 *] → [expr1 expr2 /])
5971D	(MetaDNEG)	double negate Double negation.
5976B	(MetaDINV)	double invert Double inversion.
597B5	(Meta*1)	mult by one Multiply by one.
5983B	(Meta^1)	pow by one Raise to power of one.
59885	(Meta1/)	div by one Divide by one.
5990F	(Meta+1-1)	add 1 subtract 1 Add one and subtract one.
596D3	(MetaRCOLCT)	restricted collection Restricted collection.
5C6D9	(Meta<-T)	move nearest right term to left Move nearest right term to the left.
5C68D	(MetaT->)	move nearest left term to right Move nearest left term to the right.
5C623	(Meta(()))	parenthesise nearest term Put parentheses over nearest term.
5C589	(Meta(<-)	include left term Include left term.
5C5D6	(Meta->))	include right term Include right term.
5BE81	(Meta<-->)	commute terms Commute terms.
5BECE	(Meta<-A)	associate left term Associate left term.

5BF53	(MetaA->)	associate right term Associate right term.
5C137	(Meta->())	remove prefix Remove prefix.
5C0B9	(Meta<-D)	delete left term Delete left term (via expansion).
5C102	(Meta<-D!)	delete left term Delete left term (above - ^ expansion).
5BFD8	(MetaD->)	delete right term Delete right term (via expansion).
5C3C2	(Meta<-M)	merge common factor on left Merge common factor on left side.
5C4CF	(MetaM->)	merge common factor on right Merge common factor on right side.
5C261	(Meta-())	double negate & remove prefix Double negate, then remove prefix.
5C204	(Meta1/())	double invert & remove prefix Double inversion, then remove prefix.
5C348	(MetaL*)	($\text{LN}(a^b) \rightarrow b*\text{LN}(a)$) Transform $\text{LN}(A^B)$ to $\text{LN}(A)*B$.
5C375	(MetaL())	($b*\text{LN}(a) \rightarrow \text{LN}(a^b)$) Transform $\text{LN}(A)*B$ to $\text{LN}(A^B)$.
5C2CE	(MetaE^)	($\text{EXP}(a*b) \rightarrow \text{EXP}(a)^b$) Transform $\text{EXP}(A*B)$ to $\text{EXP}(A)^B$.
5C31B	(MetaE())	($\text{EXP}(a)^b \rightarrow \text{EXP}(a*b)$) Transform $\text{EXP}(A)^B$ to $\text{EXP}(A*B)$.
5C670	(Meta->TRG)	change EXP to trig. fns. Change EXP to trigonometric functions.
5C53C	(MetaAF)	add fractions Add fractions.
5C845	(Meta->DEF)	define function Define function (SIN, SINH, ASIN...)
5C91D	(MetaTRG*)	expand trig fns. of a sum Expand trigonometric function of a sum.
5C73D	(Meta->()C%)	remove 1st RE, IM; CONJ Remove first RE, IM or CONJ.
5CDF2	(Meta<-Da11)	
5CEF1	(MetaD->a11)	
5CE15	(Meta<-Aa11)	
5CE4C	(MetaA->a11)	
5CFF5	(Meta<-Ma11)	
5D009	(MetaM->a11)	
5CF5A	(Meta<-Ta11)	
5CF23	(MetaT->a11)	

5CEBA	(Meta(<-all)	
5CE83	(Meta->)all)	
5CF91	(Meta->())all)	
5CFC3	(Meta->())C%all)	
5CD52	(evalcase:)	(meta → ?) Evaluates next object. If it drops current stream then continue, else SKIP next. Example: :: evalcase: Meta<-D Meta<-Daga ;
5CD7A	(revalcase:)	(meta → ?omeg) Evaluates next object for sub-expressions until current stream is not dropped by ob. Example: Meta<-Daga = :: revalcase: Meta<-D COLA RDRROP ; COLA RDRROP is there to mark successful operation.

2.12 Library and Backup Objects

2.12.1 Port Operations

0AAB2	PORTSTATUS	(#port → present? writeable? merged? #size #addr) Returns information for port.
0AB22	(PORTEND)	(#port → #addr) Gets end address of port.
0AB82	NEXTLIBBAK	(#addr → backup/library #nextaddr) Gets next library or backup.
0B409	(MERGE)	(#port →) Merges specified port. Only works for port one. Checks if wrong port number was entered.

2.12.2 Rompointers

07E50	#>ROMPTR	(#lib #cmd → ROMPTR) Creates rompointer.
08CCC	ROMPTR>#	(ROMPTR → #lib #cmd) Splits rompointer.
07E99	ROMPTR@	(ROMPTR → ob T) (ROMPTR → F) Recalls contents of rompointer.
62C19	DUPROMPTR@	(ROMPTR → ROMPTR ob T) (ROMPTR → ROMPTR F) Does DUP then ROMPTR@.

02FEF	(ROMSEC)	(ROMPTR → ?) Recalls contents of rompointer and EVAL. Generates "Undefined XLIB Error" if not found.
62A61	?>ROMPTR	(ob → ob') If ROM-WORD? and TYPECOL? then RPL@.
62A84	?ROMPTR>	(ob → ob') If <REF>TYPEROMP? and content exists <REF>INHARDROM? then return contents.
62BD8	RESOROMP	(→ ob) Recalls contents of next object in the runstream (which must be a rompointer).
07E76	(PTR>ROMPTR)	(ob → ROMPTR T) (ob → F) If the object is a library command, returns its rompointer and TRUE, if not just FALSE.
081FB	(ROMPTRDECOMP)	(ROMPTR → id T) (ROMPTR → F) If the library command exists and has a name, returns that name and TRUE, otherwise FALSE.
081E3	(PTR>ID)	(ob → id T) (ob → F) If the object is a library command and has a name, returns its name and TRUE, if not returns just FALSE.
07C18	(COMPILEID)	(id → id T) (id → ROMPTR T) (id → F) Searches id in current path, if found returns TRUE. Else searches attached libraries. If nothing was found, return FALSE.
61FB6	ROM-WORD?	(ob → flag)
61FA9	DUPROM-WORD?	(ob → ob flag)

2.12.3 Libraries

07709	TOSRRP	(# →) Attaches library to HOME directory. -- <REF>TEXT:Libraries
076AE	OFFSRRP	(# →) Detaches library from HOME directory. -- <REF>TEXT:Libraries
0778D	(ONSRRP?)	(# → flag) Returns TRUE if library is attached to HOME directory.

21C6F	XEQSETLIB	(% →) Internal ATTACH.
07638	SETHASH	(#libnum hxs →)
021DD	(ROMPOLL)	(→) Configures internal and external libraries. --
0210F	(DOROMPOLL)	<REF>TEXT:Libraries ({#libnum1 #libnum2..} →) Configures specified libraries. --
08199	(ROMPARTNAME)	<REF>TEXT:Libraries (#libnum → id T) (#libnum → F) Returns title of library as an ID, and TRUE. If library is not found, returns just FALSE.
081DE	(LIB>#)	(lib → #libnum T) Returns number of library.
08081	(ROMPART>ADDR)	(#libnum → #addr T) (#libnum → F) Recalls library address + 10 (prolog and length skipped).
080BF	(ROMPARTSIZE)	(#libnum → #nibbles-10 T) (#libnum → F) Returns size of library.
080DA	(NEXTROMPID)	(#libnum → #nextlibnum T) (#libnum → F) If specified library exists, #libnum is returned with TRUE.
08112	(GETHASH)	(#libnum → hxs_table T) (#libnum → F) Gets specified library's hash table.
08130	(GETMSG)	(#libnum → [] T) (#libnum → F) Gets specified library's message table. --
0764E	(SETMSG)	<REF>TEXT:Libraries ([\$] #libnum →) Sets message table of specified library. --
0813C	(GETLINK)	<REF>TEXT:Libraries (#libnum → hxs_table T) (#libnum → F) Gets specified library's link table.
08157	(GETCONFIG)	(#libnum → ob T) (#libnum → F)

07F86 (ROMPART) (rrp → {#lib1..#libn} T)
 (ROMPTR → #libnum)
Gets the list of libraries attached to the directory,
along with TRUE. If the argument is a rom pointer,
returns the library number of this pointer.

2.12.4 Backup Objects

081D9 BAKNAME (bak → id T)
Returns backup's name

0948E BAK>OB (bak → ob)
Gets backup object.

21674 >BAK (id ob → bak)
Creates backup object with specified name and
contents.

3 General SysRPL Entries

3.1 Stack Operations

03188	DUP	(ob → ob ob)
62CB9	DUPDUP	(ob → ob ob ob)
5E370	NDUPN	(ob #n → ob..ob #n) (ob #0 → #0)
62FB1	DUPROT	(1 2 → 2 2 1)
61380	DUPUNROT	(1 2 → 2 1 2) aka: SWAPOVER
630F1	DUPROLL	(1..n #n → 1 3..n #n 2)
61099	DUP4UNROLL	(1 2 3 → 3 1 2 3)
630DD	DUPPICK	(n..1 #n → n..1 #n n-1)
611F9	DUP3PICK	(1 2 → 1 2 2 1) aka: 2DUPSWAP
6119E	DUP#1+PICK	(n..1 #n → n..1 #n n)
5FC24	(DUP#2+PICK)	(n..1 #n → n..1 #n n+1)
031AC	2DUP	(1 2 → 1 2 1 2)
611F9	2DUPSWAP	(1 2 → 1 2 2 1) aka: DUP3PICK
63C40	2DUP5ROLL	(1 2 3 → 2 3 2 3 1)
031D9	NDUP	(1..n #n → 1..n 1..n)
03244	DROP	(1 →)
627A7	DROPDUP	(1 2 → 1 1)
63FA6	DROPNDROP	(1..n #n ob →)
6270C	DROPSWAP	(1 2 3 → 2 1)
62726	DROPSWAPDROP	(1 2 3 → 2) aka: ROT2DROP, XYZ>Y
62FC5	DROPROT	(1 2 3 4 → 2 3 1)
63029	DROPOVER	(1 2 3 → 1 2 1)
03258	2DROP	(1 2 →)
60F4B	3DROP	(1 2 3 →) aka: XYZ>
60F7E	4DROP	(1..4 →) aka: XYZW>
60F72	5DROP	(1..5 →)
60F66	6DROP	(1..6 →)
60F54	7DROP	(1..7 →)
0326E	NDROP	(1..n #n →)

62F75	#1+NDROP	(ob 1..n #n →) aka: N+1DROP
4B710	RESETDEPTH	(ob1..obn obn+1..obx #n → ob1..obn) Drops all but #n levels of the stack.
0314C	DEPTH	(1..n → 1..n #n)
6416D	UStackDepth	(→ #) The depth of the stack, similar to DEPTH.
5DE7D	reversym	(1..n #n → n..1 #n)
03223	SWAP	(1 2 → 2 1)
62747	SWAPDUP	(1 2 → 2 1 1)
6386C	SWAP2DUP	(1 2 → 2 1 2 1)
60F9B	SWAPDROP	(1 2 → 2) aka: XY>Y
62830	SWAPDROPDUP	(1 2 → 2 2)
6284B	SWAPDROPSWAP	(1 2 3 → 3 1) aka: UNROTDROP, XYZ>ZX
60F33	SWAPROT	(1 2 3 → 3 2 1) aka: UNROTSWAP, XYZ>ZYX
63C2C	SWAP4ROLL	(1 2 3 4 → 2 4 3 1) aka: XYZW>YWZX
61380	SWAPOVER	(1 2 → 2 1 2) aka: DUPUNROT
63C54	SWAP3PICK	(1 2 3 → 1 3 2 1)
62001	2SWAP	(1 2 3 4 → 3 4 1 2)
03295	ROT	(1 2 3 → 2 3 1)
62775	ROTDUP	(1 2 3 → 2 3 1 1)
62C7D	ROT2DUP	(1 2 3 → 2 3 1 3 1)
60F21	ROTDROP	(1 2 3 → 2 3) aka: XYZ>YZ
62726	ROT2DROP	(1 2 3 → 2) aka: DROPSWAPDROP, XYZ>Y
60F0E	ROTDROPSWAP	(1 2 3 → 3 2) aka: XYZ>ZY
60EE7	ROTSWAP	(1 2 3 → 2 1 3) aka: XYZ>YXZ
6112A	ROTROT2DROP	(1 2 3 → 3) aka: UNROT2DROP, XYZ>Z
62CA5	ROTOVER	(1 2 3 → 2 3 1 3)
60FBB	4ROLL	(1 2 3 4 → 2 3 4 1) aka: FOURROLL, XYZW>YZWX
62864	4ROLLDROP	(1 2 3 4 → 2 3 4)
62ECB	4ROLLSWAP	(1 2 3 4 → 2 3 1 4)
63001	4ROLLROT	(1 2 3 4 → 2 4 1 3) aka: FOURROLLROT

630A1	4ROLLOVER	(1 2 3 4 → 2 3 4 1 4)
60FD8	5ROLL	(1 2 3 4 5 → 2 3 4 5 1) aka: FIVEROLL
62880	5ROLLDROP	(1 2 3 4 5 → 2 3 4 5)
61002	6ROLL	(1..6 → 2..6 1) aka: SIXROLL
6106B	7ROLL	(1..7 → 2..7 1) aka: SEVENROLL
6103C	8ROLL	(1..8 → 2..8 1) aka: EIGHTROLL
03325	ROLL	(1..n #n → 2..n 1)
62F89	ROLLDROP	(1..n #n → 2..n)
62D45	ROLLSWAP	(1..n #n → 2..n-1 1 n)
612F3	#1+ROLL	(ob 1..n #n → 1..n ob)
61318	#2+ROLL	(a b 1..n #n → b 1..n a)
612DE	#+ROLL	(1..n+m #n #m → 2..n+m 1)
612CC	#-ROLL	(1..n-m #n #m → 2..n-m 1)
60FAC	UNROT	(1 2 3 → 3 1 2) aka: 3UNROLL, XYZ>ZXY
62CF5	UNROTDUP	(1 2 3 → 3 1 2 1)
6284B	UNROTDROP	(1 2 3 → 3 1) aka: SWAPDROPSWAP, XYZ>ZX
6112A	UNROT2DROP	(1 2 3 → 3) aka: ROTROT2DROP, XYZ>Z
60F33	UNROTSWAP	(1 2 3 → 3 2 1) aka: SWAPROT, XYZ>ZYX
60F0E	UNROTSWAPDROP	(1 2 3 → 3 2) aka: ROTDROPSWAP, XYZ>ZY
6308D	UNROTOVER	(1 2 3 → 3 1 2 1)
60FAC	3UNROLL	(1 2 3 → 3 1 2) aka: UNROT, XYZ>ZXY
6109E	4UNROLL	(1 2 3 4 → 4 1 2 3) aka: FOURUNROLL, XYZW>WXYZ
62D09	4UNROLLDUP	(1 2 3 4 → 4 1 2 3 3)
6113C	4UNROLL3DROP	(1 2 3 4 → 4) aka: XYZW>W
63015	4UNROLLROT	(1 2 3 4 → 4 3 2 1)
610C4	5UNROLL	(1 2 3 4 5 → 5 1 2 3 4) aka: FIVEUNROLL
610FA	6UNROLL	(1..6 → 6 1..5) aka: SIXUNROLL
62BC4	7UNROLL	(1..7 → 7 1..6)
63119	8UNROLL	(1..8 → 8 1..7)
6312D	10UNROLL	(1..10 → 10 1..9)

0339E	UNROLL	(1..n #n → n 1..n-1)
61353	#1+UNROLL	(ob 1..n #n → n ob 1..n-1)
61365	#2+UNROLL	(a b 1..n #n → n a b 1..n-1)
6133E	#+UNROLL	(1..n+m #n #m → n+m 1..n+m-1)
6132C	#-UNROLL	(1..n-m #n #m → n-m 1..n+m-1)
032C2	OVER	(1 2 → 1 2 1)
62CCD	OVERDUP	(1 2 → 1 2 1 1)
62D31	OVERSWAP	(1 2 → 1 1 2)
62D31	OVERUNROT	aka: OVERUNROT (1 2 → 1 1 2)
62D31	OVERUNROT	aka: OVERSWAP
63105	OVER#2+UNROLL	(1..n #n ob → ob 1..n #n)
63C90	OVER5PICK	(1 2 3 4 → 1 2 3 4 3 1)
63FBA	2OVER	(1 2 3 4 → 1 2 3 4 1 2)
611FE	3PICK	(1 2 3 → 1 2 3 1)
62EDF	3PICKSWAP	(1 2 3 → 1 2 1 3)
630B5	3PICKOVER	(1 2 3 → 1 2 3 1 3)
63C68	3PICK3PICK	(1 2 3 → 1 2 3 1 2)
35D08	(DROP3PICK)	(1 2 3 4 → 1 2 3 1)
6121C	4PICK	(1 2 3 4 → 1 2 3 4 1)
62EF3	4PICKSWAP	(1 2 3 4 → 1 2 3 1 4)
63C7C	SWAP4PICK	(1 2 3 4 → 1 2 4 3 1)
630C9	4PICKOVER	(1 2 3 4 → 1 2 3 4 1 4)
6123A	5PICK	(1 2 3 4 5 → 1 2 3 4 5 1)
6125E	6PICK	(1..6 → 1..6 1)
61282	7PICK	(1..7 → 1..7 1)
612A9	8PICK	(1..8 → 1..8 1)
032E2	PICK	(1..n #n → 1..n 1)
611A3	#1+PICK	(1..n #n-1 → 1..n 1)
611BE	#2+PICK	(1..n #n-2 → 1..n 1)
611D2	#3+PICK	(1..n #n-3 → 1..n 1)
611E1	#4+PICK	(1..n #n-4 → 1..n 1)
61184	#+PICK	(1..n+m #n #m → 1..n+m 1)
61172	#-PICK	(1..n-m #n #m → 1..n-m 1)

3.2 Temporary Environments

3.2.1 Built-in IDs and LAMs

15777	NULLID	(\rightarrow id) Null (empty) identifier.
34D30	NULLLAM	(\rightarrow lam) Puts NULLLAM in the stack.
211B4	(ID_CST)	ID CST
225A4	(ID_S)	ID S
3FACF	(ID_SKEY)	ID SKEY ID SKEY
3FAE8	(LAM_SKEY)	LAM SKEY
4AB1C	ID_X	ID X
4744F	'IDX	(\rightarrow id) Puts ID X unevaluated on the stack.
4AB59	ID_Y	ID Y
41A39	('idUserKeys)	(\rightarrow id) Puts ID UserKeys unevaluated on the stack.
41A43	(ID_UserKeys)	ID UserKeys ID UserKeys
41A5F	('idUserKeys.)	(\rightarrow id) Puts ID UserKeys.CRC unevaluated on the stack.
41A69	(ID_UserKeys.)	ID UserKeys.CRC
1576C	(CUREQ)	ID EQ
2C1FD	(ID_SIGMADAT)	ID Σ DAT
549DB	(lam'dvar)	LAM 'dvar LAM 'dvar
5127E	('IDPAR)	(\rightarrow id) Puts ID PPAR unevaluated on the stack. -- <REF>TEXT:Reserved PPAR

3.2.2 Conversion

05B15	\$>ID	(\$ \rightarrow ID)
63295	DUP\$>ID	(\$ \rightarrow \$ ID)
05AED	(ID>LAM)	(id \rightarrow lam)
05B01	(LAM>ID)	(lam \rightarrow id)

3.2.3 Temporary Environments Words

074D0	BIND	(obn..ob1 {lamn..lam1} \rightarrow) Binds n objects to n differently named lams.
074E4	DOBIND	(obn..ob1 lamn..lam1 #n \rightarrow) Binds n objects to n differently named lams.

634CF	1LAMBIND	(ob →) Binds one object to a null named lam.
634CA	DUP1LAMBIND	(ob → ob) Does DUP then <REF>1LAMBIND.
0DB0B0	~nNullBind	(obn..ob1 #n →) Binds #n objects to null named lams. 1LAM has the count, 2LAM the first object. Decompiles to :: ' NULLLAM CACHE ;
63A29	dvar1sBIND	(ob →) Binds ob to LAM 'dvar.
07497	ABND	(→) Abandons topmost temporary environment.
61CE9	CACHE	(obn..ob1 #n lam →) Binds all objects under the same name. 1LAM has the count.
61EA7	DUMP	(NULLLAM → ob1..obn #n) Inverse of CACHE. Always does garbage collection.
61D41	SAVESTACK	(→) Caches stack to SAVELAM.
40CE9	CacheStack	(→) Caches the stack using SAVESTACK if UNDO is on and Suspend is OK. If there was a previous environment caching the stack, it is abandoned first.
61F8F	undo	(→) Dumps SAVELAM.
07943	@LAM	(lam → ob T) (lam → F) Tries recalling object from lam. If successful, returns object and TRUE, otherwise returns just FALSE.
07D1B	STOLAM	(ob lam →) Tries storing object in lam. Generates "Undefined Local Name" error if lam is not found.
02FD6	(DoLam)	(lam → ob) (lam → !error!) Tries recalling object from lam, generates "Undefined Local Name" error if not found.
078E9	(FIRST@LAM)	(lam → ob T) (lam → F) @LAM for first environment only.
078F5	(NTH@LAM)	(lam #n → ob T) (lam #n → F) @LAM for nth environment only.
075A5	GETLAM	(#n → ob) Gets contents of nth topmost lam.
613B6	1GETLAM	(→ ob)
613E7	2GETLAM	(→ ob)

6140E	3GETLAM	(→ ob)
61438	4GETLAM	(→ ob)
6145C	5GETLAM	(→ ob)
6146C	6GETLAM	(→ ob)
6147C	7GETLAM	(→ ob)
6148C	8GETLAM	(→ ob)
6149C	9GETLAM	(→ ob)
614AC	10GETLAM	(→ ob)
614BC	11GETLAM	(→ ob)
614CC	12GETLAM	(→ ob)
614DC	13GETLAM	(→ ob)
614EC	14GETLAM	(→ ob)
614FC	15GETLAM	(→ ob)
6150C	16GETLAM	(→ ob)
6151C	17GETLAM	(→ ob)
6152C	18GETLAM	(→ ob)
6153C	19GETLAM	(→ ob)
6154C	20GETLAM	(→ ob)
6155C	21GETLAM	(→ ob)
075E9	PUTLAM	(ob #n →)
		Stores new contents to nth topmost lam.
615E0	1PUTLAM	(ob →)
615F0	2PUTLAM	(ob →)
61600	3PUTLAM	(ob →)
61615	4PUTLAM	(ob →)
61625	5PUTLAM	(ob →)
61635	6PUTLAM	(ob →)
61645	7PUTLAM	(ob →)
61655	8PUTLAM	(ob →)
61665	9PUTLAM	(ob →)
61675	10PUTLAM	(ob →)
61685	11PUTLAM	(ob →)
61695	12PUTLAM	(ob →)
616A5	13PUTLAM	(ob →)
616B5	14PUTLAM	(ob →)
616C5	15PUTLAM	(ob →)
616D5	16PUTLAM	(ob →)
616E5	17PUTLAM	(ob →)

616F5	18PUTLAM	(ob →)
61705	19PUTLAM	(ob →)
61715	20PUTLAM	(ob →)
61725	21PUTLAM	(ob →)
61735	22PUTLAM	(ob →)
61610	DUP4PUTLAM	(ob → ob)
		Does DUP then <REF>4PUTLAM .
634B6	1GETABND	(→ 1lamob)
		Does <REF>1GETLAM then <REF>ABND .
62DB3	1ABNDSWAP	(ob → 1lamob ob)
		Does <REF>1GETABND then SWAP.
62F07	1GETSWAP	(ob → 1lamob ob)
		Does <REF>1GETLAM then SWAP.
55288	1GETLAMSWP1+	(# → 1lamob #+1)
		Does <REF>1GETLAM then SWAP#1+.
632E5	2GETEVAL	(→ ?)
		Does <REF>2GETLAM then <REF>EVAL .
617D8	GETLAMPAIR	(#n → #n ob lam F)
		(#n → #n T)
		Gets lam contents and name (10 = 1lam, 20 = 2lam, etc.)
61745	DUPTEMPEVN	(→)
		Duplicates topmost temporary environment (clears protection word).
61745	DUPTEMPENV	(→)
		Duplicates topmost tempenv (clears protection word).
34D2B	1NULLLAM{ }	(→ { })
		Puts a list with one NULLLAM in the stack.
37DB9	(2NULLLAM{ })	(→ { })
		Puts a list with two times NULLLAM in the stack.
37B17	(3NULLLAM{ })	(→ { })
		Puts a list with three times NULLLAM in the stack.
52D26	(4NULLLAM{ })	(→ { })
		Puts a list with four times NULLLAM in the stack.
3306C	(7NULLLAM{ })	(→ { })
		Puts a list with seven times NULLLAM in the stack.
10E36	(8NULLLAM{ })	(→ { })
		Puts a list with eight times NULLLAM in the stack.

3.3 Error Handling

3.3.1 General Words

141E5	ERRBEEP	(→) Beeps.
04CE6	ERROR@	(→ #) Returns current error number.
04D0E	ERRORSTO	(# →) Stores new error number.
6383A	ERROROUT	(# →) Stores new error number and calls ERRJMP.
04D33	ERRORCLR	(→) Stores zero as new error number.
04ED1	ERRJMP	(→) Invokes error handling sub-system.
04E07	GETEXITMSG	(→ \$) Gets EXITMSG (user defined error message).
04E37	EXITMSGSTO	(\$ →) Stores \$ as EXITMSG.
1502F	DO#EXIT	(# →) Stores new error number, does <REF>AtUserStack and then <REF>ERRJMP.
15007	DO%EXIT	(% →) Same as above, but takes real number as argument.
1501B	(DOHXSEXIT)	(hxs →) Same as above functions, but input is hxs.
15048	DO\$EXIT	(\$ →) Stores string as EXITMSG, #70000 as error number, does <REF>AtUserStack and then <REF>ERRJMP
04EA4	ABORT	(→) Does <REF>ERRORCLR and <REF>ERRJMP .
04E5E	ERRSET	(→) Sets new error trap.
04EB8	ERRTRAP	(→) Error trap marker. If no error happens, still removes all temporary environments created since ERRSET.
13FE5	(SAVEERRN)	(→) Saves error number to last error.
1400E	(ERRO)	(→) Clears last error.
14039	(ERRN)	(→ #) Returns last error number.
1404C	(ERRN>HXS)	(→ hxs) Returns last error number as hxs.
14065	(ERRM)	(→ \$) Returns last error message.

04D87	JstGetTHEMESG	(# → \$) Fetches message from message table. To get a message from a library, use the formula: libnum*#100+msgnum. --
04D64	GETTHEMESG	<REF>TEXT:Libraries aka: JstGETTHEMESG (# → \$) If #70000 then does <REF>GETEXITMSG, else does <REF>JstGetTHEMESG . --
04DD7	(SPLITmsg)	<REF>TEXT:Libraries (#msg → #error #libnum) Splits message number into error and library numbers. -- <REF>TEXT:Libraries

3.3.2 Error Generating Words

04FB6	SETMEMERR	Error 001h Generates "Insufficient Memory" error.
04FC2	(SETDIRRECUR)	Error 002h Generates "Directory Recursion" error.
04FCE	(SETLAMERR)	Error 003h Generates "Undefined Local Name" error.
05016	SETROMPERR	Error 004h Generates "Undefined XLIB Name" error.
04FAA	(SETLBERR)	Error 006h Generates "Power Lost" error.
04FDA	(SETCORPORT)	Error 008h Generates "Invalid Card Data" error.
04FE6	(SETOBINUSE)	Error 009h Generates "Object In Use" error.
04FF2	(SETPORTNOTAV)	Error 00Ah Generates "Port Not Available" error.
04FFE	(SETNOROOM)	Error 00Bh Generates "No Room In Port" error.
0500A	(SETXNONEXT)	Error 00Ch Generates "Object Not In Port" error.
10F54	(NULLCHARERR)	Error 102h Generates "Can't Edit Null Char" error.
10F64	(INVFUNCERR)	Error 103h Generates "Invalid User Function" error.
10F74	(NOEQERR)	Error 104h Generates "No Current Equation" error.
10F86	SYNTAXERR	Error 106h Generates "Invalid Syntax" error.

10FE6	(LASTSTKERR)	Error 124h Generates "'Last Stack' Disabled" error.
10FF6	(LASTCMDERR)	Error 125h Generates "'Last Cmd' Disabled" error.
10FC6	NOHALTERR	Error 126h Generates "HALT Not Allowed" error.
11006	(ARGNUMERR)	Error 128h Generates "Wrong Argument Count" error.
11016	SETCIRCERR	Error 129h Generates "Circular Reference" error.
11026	(DIRARGERR)	Error 12Ah Generates "directory not allowed" error.
11036	(EMPTYDIRERR)	Error 12Bh Generates "Non-Empty Directory" error.
11046	(INVDEFERR)	Error 12Ch Generates "Invalid Definition" error.
11056	(MISLIBERR)	Error 12Dh Generates "Missing Library" error.
10F96	(SETINVPPAR)	Error 12Eh Generates "Invalid PPAR" error.
10FA6	(SETNONERAL)	Error 12Fh Generates "Non-real Result" error.
10FB6	(SETISOLERR)	Error 130h Generates "Unable To Isolate" error.
11066	(IDCONFERR)	Error 13Ch Generates "Name Conflict" error.
18CC2	SETSTACKERR	Error 201h Generates "Too Few Arguments" error.
18CB2	SETTYPEERR	Error 202h Generates "Bad Argument Type" error.
18CA2	SETSIZEERR	Error 203h Generates "Bad Argument Value" error.
18C92	SETNONEXTERR	Error 204h Generates "Undefined Name" error.
29DCC	(POSFLOWERR)	Error 301h Generates "Positive Underflow" error.
29DDC	(NEGFLOWERR)	Error 302h Generates "Negative Underflow" error.
29DEC	(OVERFLOWERR)	Error 303h Generates "Overflow" error.
29DFC	SETIVLERR	Error 304h Generates "Undefined Result" error.
29E0C	(INFRESERR)	Error 305h Generates "Infinite Result" error.
10EEA	(INVUNITERR)	Error B01h Generates "Invalid Unit" error.
10EFA	(CONSTUNITERR)	Error B02h Generates "Inconsistent Units" error.

2EC34	SetIOPARerr	Error C12h Generates "Invalid IOPAR" error.
0CBAE	(NOALARMERR)	Error D04h Generates "nonexistent alarm" error.
64190	Sig?ErrJump	(# →) Calls ERRJMP if the error number is any of {13E 123 DFF}.
15A40	ederr	(→) Error handler for applications which use savefmt1 to save the current display format. Calls <REF>rstfmt1 and then errors out.

3.4 Conditionals

3.4.1 Boolean Flags

5380E	COERCEFLAG	(T → %1) (F → %0) Converts system flag to user flag, drops current stream.
2A7CF	%0<>	(% → flag) Can be used to change a user flag into a system flag.
03A81	TRUE	(→ T)
0BBED	TrueTrue	(→ T T)
634F7	TrueFalse	(→ T F)
03AC0	FALSE	aka: TRUEFALSE (→ F)
6350B	FalseTrue	(→ F T) aka: FALSETRUE
2F934	FalseFalse	(→ F F)
0BC01	failed	(→ F T)
62103	DROPTRUE	(ob → T)
2F542	(2DROPTRUE)	(ob ob' → T)
5F657	(3DROPTRUE)	(ob1 ob2 ob3 → T)
10029	(4DROPTRUE)	(ob1..ob4 → T)
6210C	DROPFALSE	(ob → F)
62B0B	2DROPFALSE	(ob1 ob2 → F)
5F5E4	(4DROPFALSE)	(ob1..ob4 → F)
5F6B1	(5DROPFALSE)	(ob1..ob5 → F)
169A5	NDROPFALSE	(ob1..obn #n → F)
4F1D8	SWAPTRUE	(ob1 ob2 → ob2 ob1 T)
21660	SWAPDROPTRUE	(ob1 ob2 → ob2 T)

62EB7	XYZ>ZTRUE	(ob1 ob2 ob3 → ob3 T)
5DE41	(COLATTRUE)	(→ T) Puts TRUE in the stack and drops rest of current stream.
5DE55	RDROPFALSE	(→ F) Puts FALSE in the stack and drops rest of current stream.
03AF2	NOT	(flag → flag') Returns FALSE if the input is TRUE, and vice-versa.
03B46	AND	(flag1 flag2 → flag) Returns TRUE if both flags are TRUE.
03B75	OR	(flag1 flag2 → flag) Returns TRUE if either flag is TRUE.
03ADA	XOR	(flag1 flag2 → flag) Returns TRUE if flags are different.
635B0	ORNOT	(flag1 flag2 → flag) Returns FALSE if either flag is TRUE.
62C55	NOTAND	(flag1 flag2 → flag) Returns TRUE if flag1 is TRUE and flag2 is FALSE.
62C91	ROTAND	(flag1 ob flag2 → ob flag) Returns TRUE if either flag is TRUE.

3.4.2 General Tests

03B2E	EQ	(ob1 ob2 → flag) Returns TRUE if both objects are the same, i.e., they occupy the same physical space in memory. Only the addresses of the objects are tested.
635D8	2DUPEQ	(ob1 ob2 → ob1 ob2 flag) Does 2DUP then EQ.
63605	EQOR	(flag ob1 ob2 → flag') Does EQ then OR.
6303D	EQOVER	(ob3 ob1 ob2 → ob3 flag ob3) Does EQ then OVER.
635F1	EQ:	(ob → flag) EQ with the next object in the current stream.
635EC	DUPEQ:	(ob → ob flag) Does DUP then EQ:.
03B97	EQUAL	(ob1 ob2 → flag) Returns TRUE if the objects are equal (but not necessarily the same), i.e., their prologs and contents are the same.
635C4	EQUALNOT	(ob1 ob2 → flag) Returns TRUE if the objects are different.
63619	EQUALOR	(flag ob1 ob2 → flag') Does EQUAL then OR.

3.4.3 True/False Tests

61A3B	?SEMI	(T → :: ;) (F → :: <ob1> <rest> ;)
61A2C	NOT?SEMI	(T → :: <ob1> <rest> ;) (F → :: ;)
638E4	?SEMIDROP	(ob T → :: ob ;) (ob F → :: <ob1> <rest> ;)
61B72	NOT?DROP	(ob T → :: ob <ob1> <rest> ;) (ob F → :: <ob1> <rest> ;)
62F1B	?SWAP	(ob1 ob2 T → :: ob2 ob1 <ob1> <rest> ;) (ob1 ob2 F → :: ob1 ob2 <ob1> <rest> ;)
62D9F	?SKIPSWAP	(ob1 ob2 T → :: ob1 ob2 <ob1> <rest> ;) (ob1 ob2 F → :: ob2 ob1 <ob1> <rest> ;)
62F5C	?SWAPDROP	(ob1 ob2 T → :: ob1 <ob1> <rest> ;) (ob1 ob2 F → :: ob2 <ob1> <rest> ;)
62F43	NOT?SWAPDROP	(ob1 ob2 T → :: ob2 <ob1> <rest> ;) (ob1 ob2 F → :: ob1 <ob1> <rest> ;)
070FD	RPIT	(T ob → :: ob <ob1> <rest> ;) (F ob → :: <ob1> <rest> ;) ob is actually executed, and not pushed in the stack.
070C3	RPITE	(T ob1 ob2 → :: ob1 <ob1> <rest> ;) (F ob1 ob2 → ob2 <ob1> <rest> ;) ob1 or ob2 is actually executed, and not pushed in the stack.
61A8E	COLARPITE	(T ob1 ob2 → :: ob1 ;) (F ob1 ob2 → :: ob2 ;) ob1 or ob2 is actually executed, and not pushed in the stack.
61AE9	2'RCOLARPITE	Return to composite and ITE there.
619BC	IT	(T → :: <ob1> <rest> ;) (F → :: <ob2> <rest> ;)
0712A	?SKIP	(T → :: <ob2> <rest> ;) (F → :: <ob1> <rest> ;) aka: NOT_IT
61AD8	ITE	(T → :: <ob1> <ob3> <rest> ;) (F → :: <ob2> <rest> ;)
6381C	COLAITE	(T → :: <ob1> ;) (F → :: <ob2> ;)
61A58	ITE_DROP	(ob T → :: <ob2> <rest> ;) (ob F → :: ob <ob1> <rest> ;)
63E61	ANDITE	(f1 f2 → :: <ob1> <ob3> <rest> ;) (f1 f2 → :: <ob2> <rest> ;)
61993	case	(T → :: <ob1> ;) (F → :: <ob2> <rest> ;)

619AD	NOTcase	(T → :: <ob2> <rest> ;) (F → :: <ob1> ;)
63CEA	ANDcase	(f1 f2 → :: <ob1> ;) (f1 f2 → :: <ob2> <rest> ;)
63DDF	ANDNOTcase	(f1 f2 → :: <ob1> ;) (f1 f2 → :: <ob2> <rest> ;)
629BC	ORcase	(f1 f2 → :: <ob1> ;) (f1 f2 → :: <ob2> <rest> ;)
618F7	casedrop	(ob T → :: <ob1> ;) (ob F → :: ob <ob2> <rest> ;)
618E8	NOTcasedrop	(ob T → :: ob <ob2> <rest> ;) (ob F → :: <ob1> ;)
6191F	case2drop	(ob1 ob2 T → :: <ob1> ;) (ob1 ob2 F → :: ob1 ob2 <ob2> <rest> ;)
61910	NOTcase2drop	(ob1 ob2 T → :: ob1 ob2 <ob2> <rest> ;) (ob1 ob2 F → :: <ob1> ;)
6194B	caseDROP	(ob T → :: ;) (ob F → :: ob <ob1> <rest> ;)
61960	NOTcaseDROP	(ob T → :: ob <ob1> <rest> ;) (ob F → :: ;)
638B2	casedrptru	(ob T → T) (ob F → :: ob <ob1> <rest> ;)
6356A	casedrpf1s	Note: should be called caseDRPTRU. (ob T → F) (ob F → :: ob <ob1> <rest> ;)
63AEC	NOTcsdrpf1s	Note: should be called caseDRPF1S. (ob T → :: ob <ob1> <rest> ;) (ob F → F)
61970	case2DROP	Note: should be called NOTcaseDRPF1S. (ob1 ob2 T → :: ;) (ob1 ob2 F → :: ob1 ob2 <ob1> <rest> ;)
61984	NOTcase2DROP	(ob1 ob2 T → :: ob1 ob2 <ob1> <rest> ;) (ob1 ob2 F → :: ;)
63583	case2drpf1s	(ob1 ob2 T → F) (ob1 ob2 F → :: ob1 ob2 <ob1> <rest> ;)
634E3	caseTRUE	Note: should be called case2DRPF1S. (T → T) (F → :: <ob1> <rest> ;)
638CB	NOTcaseTRUE	(T → :: <ob1> <rest> ;) (F → T)
6359C	caseFALSE	(T → F) (F → :: <ob1> <rest> ;)
5FB49	NOTcaseFALSE	(T → :: <ob1> <rest> ;) (F → F)

62986	COLAcase	(T → :: <ob1> ;) (F → :: <ob2> <rest> ;) Drops the rest of current stream and executes case in the stream above.
629A1	COLANOTcase	(T → :: <ob2> <rest> ;) (F → :: <ob1> ;) Drops the rest of current stream and executes NOTcase in the stream above.

3.4.4 Binary Integer Tests

6336C	#=?SKIP	(#m #n → :: <ob2> <rest> ;) (#m #n → :: <ob1> <rest> ;)
63399	#>?SKIP	(#m #n → :: <ob1> <rest> ;) (#m #n → :: <ob2> <rest> ;)
62C2D	#=ITE	(#m #n → :: <ob1> <ob3> <rest> ;) (#m #n → :: <ob2> <rest> ;)
63E9D	#<ITE	(#m #n → :: <ob1> <ob3> <rest> ;) (#m #n → :: <ob2> <rest> ;)
63EB1	#>ITE	(#m #n → :: <ob2> <rest> ;) (#m #n → :: <ob1> <ob3> <rest> ;)
6186C	#=case	(#m #n → :: <ob1> ;) (#m #n → :: <ob2> <rest> ;)
6187C	OVER#=case	(#m #n → :: #m <ob1> ;) (#m #n → :: #m <ob2> <rest> ;)
618D3	#=casedrop	(#m #n → :: <ob1> ;) (#m #n → :: #m <ob2> <rest> ;) Note: should be called OVER#=casedrop.
63547	#=casedrpfls	(#m #n → F) (#m #n → :: #m <ob1> <rest> ;) Note: should be called OVER#=caseDRPFLS.
63D3A	#<>case	(#m #n → :: <ob2> <rest> ;) (#m #n → :: <ob1> ;)
63D12	#<case	(#m #n → :: <ob1> ;) (#m #n → :: <ob2> <rest> ;)
63D67	#>case	(#m #n → :: <ob2> <rest> ;) (#m #n → :: <ob1> ;)
61A18	#0=?SEMI	(#0 → :: ;) (# → :: <ob1> <rest> ;)
6333A	#0=?SKIP	(#0 → :: <ob2> <rest> ;) (# → :: <ob1> <rest> ;)
63E89	#0=ITE	(#0 → :: <ob1> <ob3> <rest> ;) (# → :: <ob2> <rest>)
63E48	DUP#0=IT	(#0 → :: #0 <ob1> <rest> ;) (# → :: # <ob2> <rest> ;)

63EC5	DUP#0=ITE	(#0 → :: #0 <ob1> <ob3> <rest> ;) (# → :: # <ob2> <rest> ;)
61896	#0=case	(#0 → :: <ob1> ;) (# → :: <ob2> <rest> ;)
61891	DUP#0=case	(#0 → :: #0 <ob1> ;) (# → :: # <ob2> <rest> ;)
618A8	DUP#0=csedrp	(#0 → :: <ob1> ;) (# → :: # <ob2> <rest> ;)
63CBD	DUP#0=csDROP	(#0 → :: ;) (# → :: # <ob1> <rest> ;)
63D26	#1=case	(#1 → :: <ob1> ;) (# → :: <ob2> <rest> ;)
63353	#1=?SKIP	(#1 → :: <ob2> <rest> ;) (# → :: <ob1> <rest> ;)
63D4E	#>2case	(#0/#1/#2 → :: <ob2> <rest> ;) (# → :: <ob1> ;)
3FF1B	?CaseKeyDef	(# #' → :: ' ob1 T ;) (# #' → :: <ob2> <rest> ;) Compares two bints. If equal, quotes the next object from the runstream and returns it along with TRUE.
3FF48	?CaseRomptr@	(# #' → ob T) (# #' → F) (# #' → :: <ob2> <rest> ;) Compares two bints. If equal, tries to resolve the rompointer which must be the next object in the runstream. The ROMPTR@ pushes TRUE when successful, so this entry can be used directly for key handlers.

3.4.5 Real and Complex Number Tests

5F127	%0=case	(%0 → :: %0 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
63D7B	j%0=case	(%0 → :: <ob1> ;) (ob → :: <ob2> <rest> ;)
5F13B	C%0=case	(C%0 → :: C%0 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
5F0FA	num0=case	(0 → :: 0 <ob1> ;) (ob → :: ob <ob2> <rest> ;) Both a real and a complex zero are TRUE conditions for this test.
5F181	%1=case	(%1 → :: %1 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
5F19F	C%1=case	(C%1 → :: C%1 <ob1> ;) (ob → :: ob <ob2> <rest> ;)

5F154	num1=case	(1 → :: 1 <ob1> ;) (ob → :: ob <ob2> <rest> ;) Both a real and a complex one are TRUE conditions for this test.
5F1EA	%2=case	(%2 → :: %2 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
5F208	C%2=case	(C%2 → :: C%2 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
5F1BD	num2=case	(2 → :: 2 <ob1> ;) (ob → :: ob <ob2> <rest> ;) Both a real and a complex two are TRUE conditions for this test.
5F267	%-1=case	(%-1 → :: %-1 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
5F285	C%-1=case	(C%-1 → :: C%-1 <ob1> ;) (ob → ob <ob2> <rest> ;)
5F23A	num-1=case	(-1 → :: -1 <ob1> ;) (ob → :: ob <ob2> <rest> ;) Both a real and a complex -1 are TRUE conditions for this test.
5EEDB	(REALNEGcase)	(%<0 → :: % <ob1> ;) (ob → :: ob <ob2> <rest> ;)
5FB6E	(pick1#0=case)	(#0 M → COLA) (ob M → SKIP)

3.4.6 Meta Object Tests

5EFD9	MEQ1stcase	(meta&ob1 ob2 → ob1=ob2 ? case) Meta&ob1 ob2 ob1=ob2 ? case
5EF15	AEQ1stcase	(meta&ob → ob=nob ? case) Meta&ob ob=nob ? case
5EFF9	MEQopscase	(meta1&ob1 meta2&ob2 ob3 →) Meta1&ob1 Meta2&ob2 ob3
5F048	AEQopscase	meta1&ob1 meta2&ob2 Meta1&ob1 Meta2&ob2
5F061	Mid1stcase	(meta&ob → ob is id) lam ? case Meta&ob ob is id or lam ? case
549EC	(MetaConccase)	(meta → meta) (Meta Meta) COLA if meta contains no ids, lams, syms or romptrs. Else SKIP.
5EF2E	(M1st+case)	Meta&+ ? case
5EF41	(M1st-case)	Meta&+ ? case Meta&- ? case
5EF54	(M1st*case)	Meta&- ? case Meta&* ? case Meta&* ? case

5EF67	(M1st/case)	Meta&/ ? case Meta&/ ? case
5EFA0	(M1st^case)	Meta&^ ? case Meta&^ ? case
58ADE	(M-1potcase)	Meta&-1&^ ? case Meta&-1&^ ? case
5EFB3	(M1stSQcase)	Meta&SQ ? case Meta&SQ ? case
5EF7A	(M1stNEGcase)	Meta&NEG ? case Meta&NEG ? case
5EF8D	(M1stINVcase)	Meta&INV ? case Meta&INV ? case
5EFC6	(M1stFNCcase)	Meta&FNCAPPLY ? case Meta&FCNAPPLY ? case
5EE10	M-1stcasechs	(Meta&NEG → Meta COLA) (Meta → Meta SKIP) (Meta&(%<0) → Meta&ABS(%) COLA) Meta&NEG Meta COLA ; Meta Meta SKIP Meta&(%<0) Meta&ABS(%) COLA

3.4.7 General Object Tests

63E2F	EQIT	(ob1 ob1 → :: <ob1> <rest> ;) (ob1 ob2 → :: <ob2> <rest> ;)
63E75	EQITE	(ob1 ob1 → :: <ob1> <ob3> <rest> ;) (ob1 ob2 → :: <ob2> <rest> ;)
63CD6	jEQcase	(ob1 ob1 → :: <ob1> ;) (ob1 ob2 → :: <ob2> <rest> ;)
61933	EQcase	(ob1 ob1 → :: ob1 <ob1> ;) (ob1 ob2 → :: ob1 <ob2> <rest> ;) Note: Should be called OVEREQcase.
629D0	REQcase	(ob → :: ob <ob2> ;) (ob → :: ob <ob3> <rest> ;) EQcase with the next object in the runstream.
618BA	EQcasedrop	(ob1 ob1 → :: <ob1> ;) (ob1 ob2 → :: ob1 <ob2> <rest> ;) Note: should be called OVEREQcasedrop.
629E9	REQcasedrop	(ob → <ob2> ;) (ob → <ob3> <rest> ;) EQcasedrop with the next object in the runstream.
63CFE	EQUALcase	(ob1 ob1 → :: <ob1> ;) (ob1 ob2 → :: <ob2> <rest> ;)
63DF3	EQUALNOTcase	(ob1 ob1 → :: <ob2> <rest> ;) (ob1 ob2 → :: <ob1> ;)
63CA4	EQUALcasedrp	(ob ob1 ob2 → :: <ob1> ;) (ob ob1 ob2 → :: ob <ob2> <rest> ;)

517FE	EQUALcasedro	(ob1 ob2 → :: <ob1> ;) (ob1 ob2 → :: ob1 <ob2> <rest> ;) Note: should be called OVEREQUALcasedrp.
517F3	EQUALcasedrop	(ob1 ob2 → :: <ob1> ;) (ob1 ob2 → :: ob1 <ob2> <rest> ;)
2856C	tok=casedrop	(\$ \$' → :: <ob1> ;) (\$ \$' → :: \$ <ob2> <rest> ;) Note: should be called OVERTok=casedrop.
5E984	nonopcase	(seco → :: seco <ob2> <rest> ;) (ob → :: ob <ob1> ;)
5F0AA	idntcase	(id → :: id <ob1> ;) (ob → :: ob <ob2> <rest> ;)
63E07	dIDNTNcase	(id → :: id <ob2> <rest> ;) (ob → :: ob <ob1> ;)
5F0CD	idntlamcase	(id/lam → :: id <ob1> ;) (ob → :: ob <ob2> <rest> ;)
63D8F	REALcase	(% → :: <ob1> ;) (ob → :: <ob2> <rest> ;)
63E1B	dREALNcase	(% → :: % <ob2> <rest> ;) (ob → :: ob <ob1> ;)
63DA3	dARRAYcase	([] → :: [] <ob1> ;) (ob → :: ob <ob2> <rest> ;)
63DB7	dLISTcase	({} → :: {} ob1 ;) (ob → :: ob <ob2> <rest> ;)
27244	NOTLISTcase	({} → :: {} <ob2> <rest> ;) (ob → :: ob <ob1> ;)
18E45	(DNOTSYMB?SEMI)	(symb → :: symb <ob1> <rest> ;) (ob → :: ob ;)
27254	NOTSECOcase	(seco → :: seco <ob2> <rest> ;) (ob → :: ob <ob1> ;)
27264	NOTROMPcase	(romp → :: romp <ob2> <rest> ;) (ob → :: ob <ob1> ;)
27224	(DNOTBAKcase)	(bak → :: bak <ob2> <rest> ;) (ob → :: ob <ob1> ;)
27234	(DNOTLIBcase)	(lib → :: lib <ob2> <rest> ;) (ob → :: ob <ob1> ;)
5EDFC	numb1stcase	(%/C%/ []/[L] → :: <ob1> ;) (ob → :: ob2 <rest> ;) If %, C%, [] or [L] then <REF>COLA, else <REF>SKIP .

3.4.8 Miscellaneous

63ED9	UserITE	(#set → :: <ob1> <ob3> <rest> ;) (#clr → :: <ob2> <rest> ;)
-------	---------	--

63EED	SysITE	(#set → :: <ob1> <ob3> <rest> ;) (#clr → :: <ob2> <rest> ;)
63BEB	caseDoBadKey	(T → :: DoBadKey ;) (F → :: <ob1> <rest> ;)
63BD2	caseDrpBadKy	aka: caseDEADKEY (ob T → :: DoBadKey ;) (ob F → :: ob <ob1> <rest> ;)
63169	caseERRJMP	(T → :: ERRJMP ;) (F → :: <ob> <rest> ;)
63B05	caseSIZEERR	(T → :: SIZEERR ;) (F → :: <ob> <rest> ;)
63B19	NcaseSIZEERR	(T → :: <ob> <rest> ;) (F → :: SIZEERR ;)
63B46	NcaseTYPEERR	(T → :: <ob1> <rest> ;) (F → :: TYPEERR ;)
40D93	NoEdit?case	(→ :: <ob1> <rest> ;) (→ :: <rest> ;)
63DCB	EditExstCase	Tests if there is no edit line active. (→ :: <ob1> <rest> ;) (→ :: <rest> ;) Tests if there is an edit line active.

3.5 Runstream Control

06E8E	NOP	(→) Does nothing.
1ACDD	xNEGNEG	(→) Does nothing, decompiles to :: CK1&Dispatch BINT0 NOP ;
06EEB	'R	(→ ob) Pushes next object in return stack (i.e., the first object in the composite above this one) to the stack (skipping it). If top return stack is empty (contains SEMI), a null secondary is pushed and the pointer is not advanced.
06F66	'REVAL	(→ ?) Does <REF>'R then <REF>EVAL.
639DE	'R'R	(→ ob1 ob2) Does <REF>'R twice.
61B89	ticR	(→ ob T) (→ F) Pushes next object in return stack to stack and TRUE, of just FALSE if the top return stack body is empty. In this case, it is dropped.

639FC	'RRDROP	(→ ob) Does <REF>'R , then <REF>RDROP.
06F9F	>R	(:: →) Pushes :: to top of return stack (skips prolog, i.e., the composite will be executed automatically).
0701F	R>	(→ ::) Creates and pops a secondary from top return stack body to stack.
07012	R@	(→ ::) Like <REF>R>, but the return stack is not popped.
0716B	IDUP	(→) Pushes interpreter pointer into the return stack.
06F8E	EVAL	(ob →) Evaluates object.
18EBA	COMPEVAL	(comp →) EVAL just pushes a list back, this one executes it.
61B45	2@REVAL	(→) EVAL first object in the stream above the previous one.
61B55	3@REVAL	(→) EVAL first object in the stream above the stream above the previous one.
619CB	GOTO	(→) Jumps to next address in stream. Address is a five-nibble address, not a system binary. Can only be used to jump to the middle of programs, cannot jump to a program prolog.
619E0	?GOTO	(flag →) If TRUE, jumps, else skips five nibbles.
619F3	NOT?GOTO	(flag →) If FALSE jumps, else skips five nibbles.
14EA5	RDUP	(→) Duplicates top return stack level.
06FB7	RDROP	(→) Pops the return stack.
6114E	2RDROP	(→) Pops two return stack levels.
61160	3RDROP	(→) Pops three return stack levels.
632F9	DROPRDROP	(ob →) Does DROP then <REF>RDROP .
62958	RDROPCOLA	(→) Does <REF>RDROP then <REF>COLA .
54C4F	(RDROPCOLATRUE)	(→ T) Does <REF>RDROP then <REF>COLATRUE .

60EBD	RSWAP	(→) Swap in the return stack.
14F2A	(RROLL)	(#n →) Rolls nth return stack level to top of return stack.
63880	RSKIP	(→) Skips first object in the return stack (i.e., the first object in the composite above this one).
0312B	SEMI	(→) DROP the rest of the current stream.

3.5.1 Quoting Objects

06E97	'	(→ nob (nextob)) Pushes next object in the stream to the stack (skipping it).
2349C	xSILENT'	(→ nextob) Put the next ob in the runstream on the stack. Quoter used in UserRPL.
63925	DUP'	(ob → ob nob) Does DUP then '.
6394D	DROP'	(ob → nob) Does DROP then '.
63939	SWAP'	(ob1 ob2 → ob2 ob1 nob) Does SWAP then '.
63961	OVER'	(ob1 ob2 → ob1 ob2 ob1 nob) Does OVER then '.
63975	STO'	(ob id/lam → nob) Does STO then '.
63989	TRUE'	(→ T nob) Pushes TRUE and the next object to the stack.
639B6	FALSE'	(→ F nob) Pushes FALSE and the next object to the stack.
6399D	ONEFALSE'	(→ #1 F nob) Pushes ONE, FALSE and the next object to the stack.
639CA	#1+'	(# → #+1 nob) Does #1+ then '.
632BD	'NOP	(→ NOP) Pushes NOP to the stack.
63155	'ERRJMP	(→ ERRJMP) Pushes ERRJMP to the stack.
3A9B8	'DROPFALSE	(→ DROPFALSE) Pushes DROPFALSE to the stack.
3FDFE	'DoBadKey	(→ DoBadKey) Pushes DoBadKey to the stack.
3FE12	'DoBadKeyT	(→ DoBadKey T) Pushes <REF>DoBadKey and TRUE to the stack.

4D11E	DROPDEADTRUE	(ob \rightarrow DoBadKey T) Makes the user drop dead, then pushes TRUE.
63B5A	'x*	(\rightarrow x*) Pushes <REF>x* (User word *) to the stack.
63B6E	'xDER	(\rightarrow xDER) Pushes xDER (User word ∂) to the stack.
5129C	'IDFUNCTION	(\rightarrow xFUNCTION) Pushes xFUNCTION (User word FUNCTION) to the stack.
512C4	'IDPOLAR	(\rightarrow xPOLAR) Pushes xPOLAR (User word POLAR) to the stack.
512B0	('IDCONIC)	(\rightarrow xCONIC) Pushes xCONIC (User word CONIC) to the stack.
512D8	'IDPARAMETER	(\rightarrow xPARAMETRIC) Pushes xPARAMETRIC (user word PARAMETRIC) to the stack.
512EC	('IDTRUTH)	(\rightarrow xTRUTH) Pushes xTRUTH (user word TRUTH) to the stack.
51300	('IDSCATTER)	(\rightarrow xSCATTER) Pushes xSCATTER (user word SCATTER) to the stack.
51314	('IDHISTOGRAM)	(\rightarrow xHISTOGRAM) Pushes xHISTOGRAM (user word HISTOGRAM) to the stack.
51328	('IDBAR)	(\rightarrow xBAR) Pushes xBAR (user word BAR) to the stack.
2520A	'Rapndit	(meta ob1...ob4 \rightarrow meta&ob ob1...ob4) Takes ob from runstream and appends it to the meta starting in level 5.
63A56	'xDEREQ	(ob \rightarrow flag) Is ob eq to user command xDER?

3.5.2 Skipping Objects

06FD1	COLA	Evals next obj and drops rest of this stream.
63A15	ONECOLA	Does ONE, then COLA.
63312	SWAPCOLA	Does SWAP, then COLA.
63326	XYZ>ZCOLA	Does UNROT2DROP, then COLA.
61A6D	COLA_EVAL	Returns and evals first obj in previous stream.
6296D	COLACOLA	Drops rest of current stream does COLA in the above one.
0714D	SKIP	Skips 1 obj in the runstream.
0715C	(2SKIP)	Skips 2 objs in the runstream.

283D8	(3SKIP)	Skips 3 objs in the runstream.
21C47	(MEMSKIP)	(ob → #nextaddress)
626EE	skipcola	Does SKIP, then COLA.
626E5	2skipcola	Does 2SKIP, then COLA.
626DC	3skipcola	Does 3SKIP, then COLA.
626AE	5skipcola	Skips 5 objects, then does COLA.
633B2	COLASKIP	Drops rest of current stream and skips one obj in above stream.
283C4	(COLAskipcola)	Drops rest of current stream, skipcola in the above.

3.6 Loops

3.6.1 Indefinite Loops

0716B	IDUP	(→) Pushes interpreter pointer into the return stack.
071A2	BEGIN	(→) Pushes interpreter pointer into the return stack.
071AB	AGAIN	(→) Sets the interpreter pointer to the topmost value in the return stack, without popping it.
071E5	REPEAT	(→) Sets the interpreter pointer to the topmost value in the return stack, without popping it.
071C8	UNTIL	(flag →) If FALSE then <REF>AGAIN, otherwise <REF>RDROP .
633C6	NOT_UNTIL	(flag →) NOT then <REF>UNTIL .
62B6F	#0=UNTIL	(# → #) Actually, should be called DUP#0=UNTIL.
071EE	WHILE	(flag →) If TRUE does nothing, otherwise <REF>RDROP then <REF>2SKIP .
633DF	NOT_WHILE	(flag →) NOT then <REF>WHILE .
633F8	DUP#0<>WHILE	(# →) Try to guess what it does.

3.6.2 Definite Loops

073F7	DO	(#stop #start →)
-------	----	--------------------

073C3	ZERO_DO	(#stop →)
6347F	DUP#0_DO	(#stop → #stop)
073CE	ONE_DO	(#stop →)
073DB	#1+_ONE_DO	(#stop →)
63498	toLEN_DO	({ } → { })
37BCB	(ONE_DO_ARRAY)	From ONE to #elements. ([] → [])
07334	LOOP	(→)
073A5	+LOOP	(# →)
		Increments index by specified number.
63466	DROPLoop	(ob →)
6344D	SWAPLoop	(ob1 ob2 → ob2 ob1)
54CB3	(SWAPDROPLoop)	(ob1 ob2 → ob2)
07321	(STOPLoop)	(→)
		Destroys topmost loop environment.
61A47	SEMILoop	(→)
07221	INDEX@	(→ #)
		Recalls topmost loop counter value.
63411	DUPINDEX@	(ob → ob #)
63425	SWAPINDEX@	(ob1 ob2 → ob2 ob1 #)
63439	OVERINDEX@	(ob1 ob2 → ob1 ob2 ob1 #)
63790	INDEX@#-	(# → #')
63790	INDEX@-	(# → #')
07270	INDEXSTO	(# →)
		Stores new topmost loop counter value.
07249	ISTOP@	(→ #)
		Recalls topmost loop stop value.
07295	ISTOPSTO	(# →)
		Stores new topmost loop stop value.
5182F	ISTOP-INDEX	(→ #)
07258	JINDEX@	(→ #)
		Recalls second topmost loop counter value.
072AD	JINDEXSTO	(# →)
		Stores new second topmost loop counter value.
07264	JSTOP@	(→ #)
		Recalls second topmost loop stop value.
072C2	JSTOPSTO	(# →)
		Stores new second topmost loop stop value.
6400F	ExitAtLoop	(→)
		Does not exit loop immediately. Just stores zero as the stop value, so all objects until the next LOOP will be evaluated. aka: ZEROISTOPSTO

3F78C	(DUPExitAtLOOP)	(ob → ob ob)
3F7EB	(ExitAtLOOPDUP)	(ob → ob ob)
4334F	(DRPExitAtLOOP)	(ob →)

3.7 Memory Operations

3.7.1 Recalling, Storing and Purging

0797B	@	(id/lam → ob T) (id/lam → F) Basic recalling function.
62C05	DUP@	(id/lam → id/lam ob T) (id/lam → id/lam F) Does DUP then <REF>@.
62A34	SAFE@	(id/lam → ob T) (id/lam → F) For lams does <REF>@. For ids does <REF>?ROMPTR> to the ob found.
5E5EE	(SAFE@NOT)	(id → ob F) (id → T) Does <REF>SAFE@ then NOT.
62A2F	DUPSAFE@	(id/lam → id/lam ob T) (id/lam → id/lam F) Does DUP then <REF>SAFE@.
1853B	SAFE@_HERE	(id → ob F) (id → T) Same as <REF>SAFE@, but works only in the current directory.
2EA6A	Sys@	(ID → ob T) (ID → F) Switches temporarily to the HOME directory and executes @ there.
20B81	XEQRCL	(id → ob) Same as <REF>SAFE@, but errors if variable is not found. Also works for lams, but you get the wrong error.
20B9A	LISTRCL	({path id} → ob) Recalls from specified path.
5E602	(@LAMNOT)	(lam → ob F) (lam → lam T) Recalls lam contents if possible.
5E616	(ROMPTR@NOT)	(ROMPTR → ob F) (ROMPTR → ROMPTR T) Recalls contents of ROMPTR if possible.

5E5B7	(@OBNOT)	(ob → ob' F) (ob → ob T) Recalls contents if input is id, lam or ROMPTR. For other types returns TRUE.
07D27	STO	(ob id/lam →) For ids this assumes ob is not pco. If replacing some object, that object is copied to TEMPOB and pointers are updated. For lams: Errors if lam is unbound.
62A02	SAFESTO	(ob id/lam →) For ids, does <REF>?>ROMPTR to the object before storing.
2E9E6	SysSTO	(ob ID →) Switches temporarily to the HOME directory and executes <REF>STO there.
18513	XEQSTOID	(ob id/lam →) Same as <REF>SAFESTO, but will only store in the current directory and will not overwrite a directory. aka: ?STO_HERE
40F22	XEQStoKey	(ob ID →)
085D3	REPLACE	(newob oldob → newob) Replaces oldob (in memory) with newob.
08C27	PURGE	(id →) Purges variable. Does no type check first.
1854F	?PURGE_HERE	(id →) Like <REF>PURGE, but only works in current directory.
7DF7C	MOVEVAR	Move the variable to a different directory. Stack diagram unknown - level 1 must be rrp, but level two??
08696	CREATE	(ob id →) Creates a variable in the current directory. Errors if id is or contains current directory. Assumes id is not a pco.
185C7	DoHere:	(→) Next object in the runstream is evaluated for the current directory only.
63A3D	'LAMLNAMESTO	(ob →) STO to LAM LAMLNAME.

3.7.2 Directories

077E4	(MAKERRP)	(#libnum → rrp) Creates an empty directory.
-------	-----------	--

08DF2	(CREATERRP)	(id →) Creates an empty directory. Does not check if the name is already used.
		:: # 7FF CRDIR# SWAP CREATE ;
184E1	CREATEDIR	(id →) Creates an empty directory. Calls <REF>?PURGE_HERE first to delete the original.
08326	LASTRAM-WORD	(rrp → ob T) (rrp → F) Recalls first object in directory.
18621	LastNonNull	(rrp → ob T) (rrp → F) Recalls first object in directory (not null named).
08376	PREVRAM-WORD	(ob → ob' T) (ob → F) Recalls next object in directory.
1863A	PrevNonNull	(ob → ob' T) (ob → F) Recalls next object in directory (not null named).
18653	(CkNonNull)	(ob → ob T) (ob → F) Checks that the variable (ob) has a name.
082E3	RAM-WORDNAME	(ob → id) Recalls name of object in current directory.
18595	XEQPGDIR	(id →) Purges a directory. Checks references, etc. first.
20FF2	XEQORDER	({id1 id2..} →) Orders the variables in the directory by moving the given variables to the beginning of the directory.
18779	DOVARS	(→ {id1 id2..}) Returns list of variables from current directory.
186E8	DOTVARS%	(% → { }) Returns a list of variables in the current directory with user type given by the number. Internal TVARS if a single number was given.
1867F	(DODIRPRG)	(ob :: → { }) Executes seco (can be single object) on all directory variables. At execution: ob :: id_contents { } id To be returned: ob :: id_contents { } ob flag If flag is TRUE, ob is added with >TCOMP to list, else it is dropped.
1848C	PATHDIR	(→ {HOME dir1 dir2..}) Returns current path.
1A16F	UPDIR	(→) Goes to parent directory.

08309	(MYRAMROMPAIR)	(rrp → rrp' T) (rrp → F) Gets parent directory. Returns FALSE if parent directory is HOME.
08DD4	SYSRRP?	(rrp → flag) Is rrp HOME?
08D5A	CONTEXT@	(→ rrp) Recalls current directory.
08D08	CONTEXT!	(rrp →) Sets new current directory.
08DD4	SYSRRP?	(rrp → flag) Is rrp HOME?
08D82	(STOPSIGN@)	(→ rrp) Recalls last directory.
08D4A	(STOPSIGN!)	(rrp →) Stores new last directory.
08D92	HOMEDIR	(→) Sets HOME as current directory. aka: SYSCONTEXT
08DC4	(SYSSTOPSIGN)	(→) Sets HOME as last directory.
640A0	SaveVarRes	(→) Binds current and last directories to two nullnamed lams.
640FA	RestVarRes	(→) First sets HOME as both the current and last directories (in case an error happens). Then, restores the current and last directories from 1LAM and 2LAM.

3.7.3 The Hidden Directory

640BE	SetHiddenRes	(→) Sets the hidden directory as the current and last directories.
64037	WithHidden	(→ ?) Executes next command in hidden directory.
64023	RclHiddenVar	(id → ob T) (id → F) Recalls variable in hidden directory. Same as :: WithHidden @ ;
64078	StoHiddenVar	(ob id →) Stores variable in hidden directory. Same as :: WithHidden STO ;
6408C	PuHiddenVar	(id →) Purges variable in hidden directory. Same as :: WithHidden PURGE ;

3.7.4 Temporary Memory

06657	TOTEMPOB	(ob → ob') Copies object to TEMPOB and returns pointer to the new copy.
62C69	TOTEMPSWAP	(ob1 ob2 → ob2' ob1) Does TOTEMPOB then SWAP.
37B44	CKREF	(ob → ob') If object is in TEMPOB, is not embedded in a composite and not referenced, does nothing. Else copies it to TEMPOB and returns the copy.
63F7E	SWAPCKREF	(ob1 ob2 → ob2 ob1') Does SWAP then <REF>CKREF.
06B4E	INTEMNOTREF?	(ob → ob flag) If the object is in TEMPOB area, is not embedded in a composite and is not referenced, returns the object and TRUE, otherwise returns the object and FALSE.
06B3E	(FREEINTEMP?)	(ob → ob flag) Tests if object is in TEMPOB area and not in a composite.
01E0E8	~INTEMPOB?	(ob → ob flag)
065D9	(PTRREFD?)	(ob → ob flag) Tests if object is referenced.
065E5	(REFERENCED?)	(ob → ob flag) Tests if object is referenced or in composite.
06BC2	(NOTREF?)	(ob → ob flag) Tests if object is not referenced or in composite. (:: REFERENCED? NOT ;)
06DDE	(>TOPTEMP)	(ob → ob') Moves object to top ob TEMPOB area. Does not garbage collection.
064BD	(TOTEMPOBADJ)	(ob → ob ob') Makes a standalone copy by moving references to a new copy.
064D6	(DOADJ1)	(ob1 ob2 → ob1 ob') Moves references from ob2 to ob1 (ob1 in TEMPOB area).
064E2	(DOADJ)	(ob1 ob2 → ob1 ob') Moves references from ob2 to ob1 (ob1 in TEMPOB area). References to body of ob2 are moved too.

3.8 Time and Alarms

40EE7	SLOW	(→) 15 millisecond delay.
40F02	VERYSLOW	(→) 300 millisecond delay.
48FF9	SORTASLOW	(→) 1.2 second delay (4 x VERYSLOW).
40F12	VERYVERYSLOW	(→) 3 second delay.
1A7ED	(wait)	(hxs →) Wait specified number of ticks (there are 8192 ticks in a second).
1A7C9	dowait	(%secs →) Waits specified number of seconds.
1A7B5	(dowait/quit?)	(%secs →) Waits specified number of seconds, exits program if <u>CANCEL</u> is pressed.
2A673	%>HMS	(% → %hms) Converts from decimal to H.MMSS format.
2AF27	%%H>HMS	(%% → %%hms) Same as %>HMS, but for long reals.
2A68C	%HMS>	(%hms → %) Converts from H.MMSS format to decimal.
2A6A0	%HMS+	(%hms1 %hms2 → %hms) Adds time in hms format.
2A6C8	%HMS-	(%hms1 %hms2 → %hms) Subtracts time in hms format.
0CBFA	TOD	(→ %time) Returns current time.
0CD53	(>TIME)	(%time →) Sets time.
0CD3F	(CLKADJ)	(%time →) Also sets time.
0E66A	VerifyTOD	(%time → %time) Checks for validity of time. Errors if not valid.
0CC0E	DATE	(→ %date) Returns current date.
0CD2B	(>DATE)	(%date →) Sets date, errors if % is not a valid date.
0CC5B	DATE+DAYS	(%date %days → %date') Adds specified number of days to date.
0CC39	DDAYS	(%date1 %date2 → %days) Returns number of days between two dates.
0EB81	CLKTICKS	(→ hxs) Returns tick count. aka: SysTime

OD304	TIMESTR	(%dt %tm → "dy dt tm") Returns string representation of time, using current format. Example: "WED 06/24/98 10:00:45A"
OCFD9	Date>d\$	(%date → \$) Returns string representation of date, using current format.
OD2F0	(Date>wd\$)	(%date → \$weekday) Returns weekday: "SUN", "MON", etc.
OCF5B	(Ticks>wd\$)	(hxs → \$weekday) Same function but using clock ticks.
OD06A	TOD>t\$	(%time → \$) Returns string represent the time, using current format.
OEE50	Date>hxs13	(%date → hxs) Converts date to ticks.
OD156	(Ticks>Date)	(hxs → %date) Returns date from hxs of internal alarm list format.
OEE83	(TOD>Ticks)	(%time → hxs) Converts time to ticks.
OD143	(Ticks>TOD)	(hxs → %time) Returns time from hxs of internal alarm list format.
OD169	(Ticks>Rpt)	(hxs → %rpt) Converts hxs in internal alarm list format to repetition interval.
OEE26	(Date+Time)	(hxs_d hxs_t → hxs) Takes two hxs representing the date and the time, and joins them into only one hxs.

3.8.1 Alarms

OE235	ALARMS@	(→ {}) Returns internal alarms list.
OE6ED	STOALM	(%date %time acti %rep → %) Stores an alarm. %repeat is the number of ticks between every repetition. Since there are 8192 ticks in a second, 60 seconds in a minute, and 60 minutes in an hour, to make an alarm that repeats every hour, %repetition would be $8192*60*60 = 29491200$. Returns real number representing the position of the alarm in the list.

OE54D	(STOALARMLS)	({ } → %) Stores an alarm. List contents: { %date %time action %repeats } You may omit %repeats and action. In this case, the alarm has no repetition and no message is displayed. Returns real number representing the position of the alarm in the list.
OE510	(STOALARM%)	(%time → %) Store an alarm at specified time today, with no message and no repetition. Returns real number representing the position of the alarm in the list.
OE724	(PURGALARM%)	(% →) Internal <REF>xDELALARM.
0EF45	(>ALRMLS)	(\$ %date %time %rpt → { }) Generates list (of the internal type) representing the alarm.
OE1D8	(ALRMLS>)	({ } → { }') Converts list of internal format to list in the format of STOALARMLS.
OE402	(RCLALM)	(#n → { } T) (#n → F) Recalls nth alarm. List is in the internal format.
OE3DF	(RCLALARM%)	(%n → { }) Recalls nth alarm. List is in the format of STOALARMLS.
OEAD7	(FINDALARM%)	(%date → %) Returns position in the internal alarm list of the first alarm of that day (or in any following day).
OE31	(FINDALARMLS)	({ } → %) Takes a list of the format: { %date %time } Returns real represent the position of the specified alarm in the alarm list, or 0 if not found.
OE724	(DELALARM)	(%n →) Deletes nth alarm.
422A1	ALARM?	(→ flag) Returns TRUE if an alarm is due.
ODDC1	(ACKALM)	(→ flag) Tries acknowledging first alarm due. Returns TRUE if no due alarm was found, or FALSE if a due alarm has been found and acknowledged.
ODDA8	(ACKALLALMS)	(→) Acknowledges all due alarms.
OE31	(FNDALARM{ })	
OE5D5	FindNext	

3.9 System Functions

3.9.1 User and System Flags

53731	SetSysFlag	(# →) Sets the system flag with number #. <REF>TEXT:Flags
53761	ClrSysFlag	(# →) Clears the system flag with number #. <REF>TEXT:Flags
53784	TestSysFlag	(# → flag) Returns TRUE if system flag is set. <REF>TEXT:Flags
1C4EC	(TestSysClr)	(# → flag) Clears flag after testing. <REF>TEXT:Flags
3EDA2	(TogSysFlag)	(# →) Toggles system flag. <REF>TEXT:Flags
53725	SetUserFlag	(# →) Set the user flag with number #. <REF>TEXT:Flags
53755	ClrUserFlag	(# →) Clear the user flag with number #. <REF>TEXT:Flags
53778	TestUserFlag	(# → flag) Returns TRUE if user flag is set. <REF>TEXT:Flags
1C4CE	(TestUserClr)	(# → flag) Clears flag after testing. <REF>TEXT:Flags
1C637	RCLSYSF	(→ hxs) Recalls system flags from 1 to 64. <REF>TEXT:Flags
1C731	(STOSYSF)	(hxs →) Stores system flags from 1 to 64. <REF>TEXT:Flags
1C6E3	DOSTOSYSF	(hxs →) Stores system flags from 1 to 64, checking for changes in LASTARG flag.
1C64E	RCLUSERF	(→ hxs) Recalls user flags from 1 to 64.
1C6F7	(STOUSERF)	(hxs →) Stores user flags from 1 to 64.
1C6CF	(STOALLFcont)	(hxs_usr hxs_sys →) Stores user and system flags from 1 to 64. First is user flags, second is system flags.

1C6A2	DOSTOALLFO	({ } →) Stores user and system flags from 1 to 64. Expects a list of two hxs, first is user flags, second is system flags.
53C96	(XEQSTWS)	(% →)
53CF0	(XEQRCWS)	(→ %)
53C37	DOHEX	(→) Switch stack display format of HEX strings to hexadecimal. <REF>TEXT:Flags
53C5B	DODEC	(→) Switch stack display format of HEX strings to decimal. <REF>TEXT:Flags
53C43	DOBIN	(→) Switch stack display format of HEX strings to binary.
53C4F	DOOCT	(→) Switch stack display of HEX strings to octal.
54050	BASE	(→ #) Returns #10h, #10d, #10b or #10o. In decimal terms, 16 for hexadecimal base, 10 for decimal base, 8 for octal base or 2 for binary base.
5407A	(BASECHAR)	(→ char) Returns "h", "d", "b" or "o".
16707	DOSTD	(→) Internal version of user word STD.
166E3	DOFIX	(# →) Internal version of user word FIX.
166EF	DOSCI	(# →) Internal version of user word SCI.
166FB	DOENG	(# →) Internal version of user word ENG.
15A8B	savefmt1	(→) Saves the current number format, and changes to STD mode.
15A60	rstfmt1	(→) Restores the number format saved by savefmt1. Only one set of flags can be saved, there is no nesting of these entries.
53B61	(NumbMode)	(→ #) Returns 0 for STD mode, 1 for FIX mode, 2 for SCI mode or 3 for ENG mode.
2A5F0	SETRAD	(→) Set angular mode to RAD.
53BDD	RAD?	(→ flag) Is angular mode RAD?
2A5D2	SETDEG	(→) Set angular mode DEG.

53BC9	(DEG?)	(→ flag) Is angular mode DEG?
2A604	SETGRAD	(→) Set angular mode GRAD.
167BF	DPRADIX?	(→ flag) Returns TRUE if current radix is ".".
53C23	(PRSOL?)	(→ flag) Returns TRUE if general solutions flag (1) is set.
53C0A	(NOTCONST?)	(→ flag) Returns TRUE if symbolic constants flag (2) is cleared.
53B9C	(SETNUM)	(→) Sets numeric results flag (3).
53B88	(CLRNUM)	(→) Clears numeric results flag (3).
53BB0	(NOTNUM?)	(→ flag) Returns TRUE if numeric results flag (3) is cleared.
538DC	UNDO_OFF	(→) Turns saving of the last stack for UNDO off.
538CE	UNDO_ON	(→) Turns saving of the last stack for UNDO on.
538C0	UNDO_ON?	(→ flag) Tests if last stack saving for UNDO is on.
3AA0A	1A/LockA	(→) Equivalent to pressing the ALPHA key, turns on ALPHA mode for either 1 keypress or until the next ALPHA keypress, depending on system flag 60. -- Flags: -60

3.9.2 General Functions

1415A	DOBEEP	(%freq %dur →) Beeps. Analog to user function BEEP.
141B2	setbeep	(#ms #Hz →) Also beeps.
041A7	TurnOff	(→) Internal OFF.
041DA	(!TurnOff)	(→) Internal OFF. Does not do alarm check, etc.
041ED	DEEPSLEEP	(→ flag) Puts HP into deepsleep mode. Returns TRUE if "Invalid Card Data" message.
325AA	ChkLowBat	

0426A	ShowInvRomp	(→) Flashes "Invalid Card Data" message.
386D8	?FlashAlert	(→) Displays system warnings.
04544	(AlertStatus)	(→ #) Gets last system warning: #0h = OK #1h = Alarm #2h = LowBat (S) #4h = LowBat (P1) #8h = LowBat (P2)
04575	(Alert\$)	(# → \$) Recalls system warning message.
0D2A3	(WSLOG)	(→ \$4 \$3 \$2 \$1) Recalls warm start log messages.
0D18A	(WSLOGN)	(#n → \$) Recalls specified warm start log message.
21B4E	(DOAPWL)	(→) Forces a warm start but does not log a warmstart event.
04912	(LiteSlp)	(→) Enters light sleep mode.
05F42	GARBAGE	(→) Forces garbage collection.
05F61	MEM	(→ #) Returns amount of free memory in nibbles. Does not do garbage collection. (The user word does.)
05902	OSIZE	(ob → #) Returns object size in nibbles. Forces garbage collection.
05944	OCRC	(ob → #nib hxs) Returns size in nibbles and checksum as hxs.
1A1FC	OCRC%	(ob → hxs %bytes) Returns checksum and size in bytes.
1A265	VARSIZE	(id → hxs %bytes) Returns checksum and size in bytes of specified variable.
1A2DA	INHARDROM?	(ob → ob flag) Is object address < #80000h?
19350	(NOTINHARDROM?)	(ob → ob flag) Is object address #80000h?
05AB3	CHANGETYPE	(ob #prolog → ob') Changes prolog of object, does TOTEMPOB.
05ACC	(!CHANGETYPE)	(ob #prolog → ob') Changes prolog of object.

6595A	getnibs	(hxs hxs → hxs') Peek. First hxs is data, second is address. The data is overwritten for its length (maximum 16) with nibbles starting from specified address.
6594E	putnibs	(hxs hxs →) Poke. First hxs is data, second is address. See <REF>getnibs.
538F8	NOBLINK	(→) Clears the BLINKFLAG, SysNib5.
13D28	cursorblink+	
13D55	cursorblink-	
42078	?BlinkCursor	(→) Makes the cursor Blink if in App-mode or Editline.

3.10 Kermit

2D396	MAXRETRY	Maximum number of retries = 10.
2D3A0	LAMPKNO	Contains packet number.
2D3B1	LAMPACKET	
2D3C6	LAMRETRY	Contains #retry.
2D3D9	(LAMERRMSG)	
2D3EE	LAMKP	
2D3FB	LAMLNAME	
2D40E	LAMOBJ	
2D41D	LAMOPOS	
2D42E	(LAMEXCHP)	
2D45A	LAMKLIST	
2D46D	LAMKMODE	
2D480	(LAMKPTRN)	
2D493	LAMKRM	
2D4A2	(LAMMaxR)	
2F211	LAMKML	
2D441	'LamKPSto	
2D816	(DORECN)	(\$/id/lam →) Internal RECN.
2D9F5	(DOSERVER)	Internal SERVER.
2E6EB	SENDLIST	({} →) Internal SEND.

2E5AB	(SENDNAME)	(id/lam →) Internal SEND.
2E7EF	GETNAME	(\$/id/lam →) Internal KGET.
2E876	DOFINISH	(→) Internal FINISH.
2E8D1	DOPKT	(\$ \$' →) Internal PKT.
2EC84	DOBAUD	(% →) Internal BAUD.
2ECCA	DOPARITY	(% →) Internal PARITY.
2ED10	DOTRANSIO	(% →) Internal TRANSIO.
2ED4C	(DOCKSM)	Internal CKSM.
2EDA6	DOKERM	(→ \$) Internal KERM.
2EDE1	DOBUFLEN	(→ % 0/1) Internal BUFLen.
2EDF5	(DOSTIME)	Internal STIME.
2EE18	DOSBRK	(→) Internal SBRK.
2EE6F	(DOXMIT)	(\$ →) Internal XMIT.
2EE97	DOSRECV	(% →) Internal SRECV.
315C6	CLOSEUART	(→) Internal CLOSEIO.
31868	DOCR	(→) Internal CR.
31FFD	DODELAY	(% →) Internal DELAY.
2D730	KDispRow2	
2D74E	KDispStatus2	
2D517	EXCHINITPK	
2D58C	SENDEOT	
2E0A9	SENDNAK	
2E0C7	SENDERROR	
2E0F4	SENDPKT	
2E6BE	InitIOEnv	
2EAE2	KERMOPEN	
2EB37	DOOPENIO	
2EB62	OpenIO	
3187C	OpenIOPrt	

2E4DC	APNDCRLF	(\$ → \$') Appends carriage return and line feed to string.
31854	docr	
2EC25	IOCheckReal	
2E99E	StdIOPAR	(→ {}) Default IOPAR: { 9600 0 0 0 3 1 }.
2EA4F	GetIOPAR	(→ %baud % % % %) Recalls IOPAR and explodes it into the stack.
2E9CB	StoIOPAR	({} →) STO the list of IO parameters in the HOME directory in the variable IOPAR.
2EC34	SetIOPARerr	Error C12h Generates "Invalid IOPAR" error.
31F4A	StdPRTPAR	
31F7D	StoPRTPAR	
3205C	GetChkPRTPAR	
30ED2	OUTUART	
31289	POPUART	
3161E	OpenUartClr	
315F9	CloseUart	
31608	(OpenUart?Clr)	
32161	PRINT	
32387	PRINTxNLF	
323F9	REMAP	
3252B	SetEcma94	
324A6	AllowPrlcdCl	
32B74	PrintGrob	
2D9A1	SetServMode	
2D9B2	ClrServMode	
2EEC4	SendSetup	
2EFD7	TRPACKETFAIL	
2F383	IncrLAMPKNO	Increases packet number.
2F39C	GetKermPkt#	
2F989	RecvNextPkt	

2FEC9	KVISLF	(\$ → \$') String translation for transfer from HP to PC. Inserts <cr> (character 12) in front of every newline (character 10), and translates characters >127 to the corresponding backslash escape. Which translations are being made depends upon the current translation mode (the last number in the IOPAR variable, can be set with DOTRANSIO). 0: No translation 1: CRLF translation 2: CRLF and characters 128-159 (80h-9Fh) 3: CRLF and characters 128-255 (80h-FFh)
2FEDD	KVIS	(\$ → \$') Like <REF>KVISLF, but never translates newlines.
3016B	KINVISLF	(\$ → \$' \$') String translation for transfer from PC to HP. Translates digraphs in the string to characters and removes <cr> (character 12) in front of newline characters. Which translations are actually made depends upon the current translation mode, see KVISLF. '\$' contains any incomplete trailing backslash sequence in the original string.
307E2	GETKP	
2FEA1	SENDACK	
2FEB5	SENDNULLACK	
30914	ACK_INIT	
30B1D	(CHOOSE_INIT)	
30BBE	ENCODE1PKT	
30BD7	ENCODE	
30D31	DECODE	
3133B	UARTBUFLLEN	
3136C	FLUSHRSBUF	
31444	PUTSERIAL	
314E5	GETSERIAL	
30794	VERSTRING	(→ \$) Returns version string.
3037A	(VERSTRING?)	(\$ → \$ flag) Checks if \$ starts with VERSTRING.
42249	UART?	
42145	UARTxcp	
2D5E1	SEND_PACKET	
21B5A	XEQIOBACKUP	

00C10	kermpktmsg
00C0E	kermrecvmsg
00C0D	kermsendmsg

4 Input and Output

4.1 Checking for Arguments

4.1.1 Number and Type of Arguments

18A1E	CK0	(→) Saves current command to LASTCKCMD. Marks stack below level 1 to STACKMARK.
18AA5	CK1	(ob → ob) Saves current command to LASTCKCMD. Verifies that there is at least one object in the stack, if not generates a "Too Few Arguments" error. Saves stack mark to STACKMARK. If Last Arg is enabled then saves the argument.
18A80	CK2	(ob1 ob2 → ob1 ob2) Like <REF>CK1, but checks for at least two arguments.
18A5B	CK3	(ob1...ob3 → ob1...ob3) Like <REF>CK1, but checks for at least three arguments.
18B92	CK4	(ob1...ob5 → ob1...ob5) Like <REF>CK1, but checks for at least four arguments.
18B6D	CK5	(ob1...ob5 → ob1...ob5) Like <REF>CK1, but checks for at least five arguments.
18C34	CKN	(ob1...obn %n → ob1..obn #n) Checks for a real in level one. Then checks for that number of arguments. Finally, converts the real to a bint.
18A15	CKONOLASTWD	(→) Like <REF>CK0, but does not save current command.
18AB2	CK1NOLASTWD	(ob → ob) Like <REF>CK1, but does not save current command.
18A8D	CK2NOLASTWD	(ob1 ob2 → ob1 ob2) Like <REF>CK2, but does not save current command.
18A68	CK3NOLASTWD	(ob1...ob3 → ob1...ob3) Like <REF>CK3, but does not save current command.
18B9F	CK4NOLASTWD	(ob1...ob4 → ob1...ob4) Like <REF>CK4, but does not save current command.

18B7A	CK5NOLASTWD	(ob1...ob5 → ob1...ob5) Like <REF>CK5, but does not save current command.	
18C4A	CKNNOLASTWD	(ob1...obn %n → ob1...obn #n) Like <REF>CKN, but does not save current command.	
18F9D	CK&DISPATCH0	(→) Dispatches on stack argument. Does not convert ZINTs to REAL. -- <REF>CK&DISPATCH1 <REF>CK&DISPATCH2 <REF>TEXT:Dispatch.Types	
18FB2	CK&DISPATCH1	(→) Dispatches on stack arguments, stripping tags and converting ZINTS to REALS (HP49 only) if necessary. -- <REF>CK&DISPATCH0 <REF>CK&DISPATCH2 <REF>TEXT:Dispatch.Types	
18FA9	CK&DISPATCH2	(→) Equivalent to <REF>CK&DISPATCH1. -- <REF>CK&DISPATCH0 <REF>TEXT:Dispatch.Types	
18ECE	CK1&Dispatch	(→) Combines <REF>CK1 with <REF>CK&DISPATCH1. -- <REF>TEXT:Dispatch.Types	
18EDF	CK2&Dispatch	(→) Combines <REF>CK2 with <REF>CK&DISPATCH1. -- <REF>TEXT:Dispatch.Types	
18EF0	CK3&Dispatch	(→) Combines <REF>CK3 with <REF>CK&DISPATCH1. -- <REF>TEXT:Dispatch.Types	
18F01	CK4&Dispatch	(→) Combines <REF>CK4 with <REF>CK&DISPATCH1. -- <REF>TEXT:Dispatch.Types	

18F12	CK5&Dispatch	(→) Combines <REF>CK5 with <REF>CK&DISPATCH1. -- <REF>TEXT:Dispatch.Types
5EA09	CKINFARGS	(→) Gets meta as argument and checks its length (using DEPTH), and errors if it is too short. Collects the arguments to a list, does CK1NOLASTWD, and explodes the meta again.
1884D	OLASTOWDOB!	(→) Clears command save by last CK<n> command. <REF>CK0 aka: OLASTOWDOB!, OLastRomWrd!
40BC9	AtUserStack	(→) :: CKONOLASTWD OLASTOWDOB! ;
1592D	CK1NoBlame	(→) :: OLASTOWDOB! CK1NOLASTWD ;
62474	'RSAVEWORD	(→) Stores first object in the composite above the actual to LASTCKCMD. aka: 'RSaveRomWrd
18F23	EvalNoCK	(comp → ?) Evaluates composite without saving as current com- mand. If first command is CK<n>&Dispatch it is replaced by CK&DISPATCH1. If first command is CK<n> it is skipped. Any other first command is also skipped!
18F6A	(EvalNoCK:)	Run Stream: (ob →) <REF>EvalNoCK with the next object in the run- stream as argument.
18F6A	('EvalNoCK:_sup)	Run Stream: (ob →) <REF>EvalNoCK with the next object in the run- stream as argument. aka: EvalNoCK:

4.1.2 Type Checking

63B2D	CKREAL	(% → %) (Z → %) Checks for real. If a ZINT, convert to real. Else SETTYPEERR.
54C63	nmetasyms	(meta → meta) Checks for meta containing %, C%, unit, id, lam or symb.

19207	(CKNFLOATS)	(→) Checks for #n floats (F%) zero = C%0 if at least one float was complex, otherwise it is %0.
03C64	TYPE	(ob → #prolog) Returns address of prolog of object.
1CB90	XEQTYPE	(ob → ob %type) System version of user word TYPE, but this keeps the object.
6216E	TYPEREAL?	(ob → flag)
62169	DUPTYPEREAL?	(ob → ob flag) aka: DTYPEREAL?
62183	TYPECMP?	(ob → flag)
6217E	DUPTYPECMP?	(ob → ob flag)
62159	TYPECSTR?	(ob → flag)
62154	DUPTYPECSTR?	(ob → ob flag) aka: DTYPECSTR?
62198	TYPEARRAY?	(ob → flag)
62193	DUPTYPEARRAY?	(ob → ob flag) aka: DTYPEARRAY?
62198	TYPEARRY?	(ob → flag ???)
6223B	TYPERARRY?	(ob → flag)
62256	TYPECARRY?	(ob → flag)
62216	TYPELIST?	(ob → flag)
62211	DUPTYPELIST?	(ob → ob flag) aka: DTYPELIST?
6203A	TYPEIDNT?	(ob → flag)
62035	DUPTYPEIDNT?	(ob → ob flag)
6211A	TYPELAM?	(ob → flag)
62115	DUPTYPELAM?	(ob → ob flag)
621D7	YPESYMB?	(ob → flag)
621D2	DUPTYPESYMB?	(ob → ob flag)
62144	TYPEHSTR?	(ob → flag)
6213F	DUPTYPEHSTR?	(ob → ob flag)
62201	TYPEGROB?	(ob → flag)
621FC	DUPTYPEGROB?	(ob → ob flag)
6222B	TYPETAGGED?	(ob → flag)
62226	DUPTYPETAG?	(ob → ob flag)
6204F	TYPEEXT?	(ob → flag) Is ob a unit object?
6204A	DUPTYPEEXT?	(ob → ob flag) Is ob a unit object?
621AD	TYPEROMP?	(ob → flag)

621A8	DUPTYPEROMP?	(ob → ob flag)
6212F	TYPEBINT?	(ob → flag)
6212A	DUPTYPEBINT?	(ob → ob flag)
621C2	TYPERRP?	(ob → flag)
621BD	DUPTYPERRP?	(ob → ob flag)
62025	TYPECHAR?	(ob → flag)
62020	DUPTYPECHAR?	(ob → ob flag)
621EC	TYPECOL?	(ob → flag) Is on a secondary?
621E7	DUPTYPECOL?	(ob → ob flag) Is ob a secondary? aka: DTYPECOL?
26A2D	?OKINALG	(ob → ob flag) Is object allowed in algebraics?

4.2 Keyboard Control

4.2.1 Converting Keycodes

41CA2	Ck&DecKeyLoc	(%rc.p → #kc #p) Converts from user key representation format to system. Does handle shift-hold keys.
41D92	CodePl>%rc.p	(#kc #p → %rc.p) Converts from system key representation format to user. Does handle shift-hold keys.
3FE44	H/W>KeyCode	(# → #') Converts the keycode offset for shift keys to the keycode of the shift key, i.e. 80h->32d, 40h->37d, C0h->42d
3FE26	H/WKey>KeyOb	
3FE7B	ModifierKey?	(#kc #pl → flag) Is the key any of the three modifiers right-shift, left-shift, or alpha?
40A82	KeyOb@	(→ id/romptr) Returns the object assigned the the key which caused the current program to be executed, or whatever has been stored with KeyOb!
40A6F	KeyOb!	(ob →) Store ob as the KeyOb.
40A95	KeyOb0	(→) Clear the KeyOb.

4.2.2 Waiting for Keys

00D71	FLUSHKEYS	(→) Flushes the key buffer. aka: FLUSH
04708	CHECKKEY	(→ #kc T) (→ F) Returns next key in the key buffer (if there is one), but does not pop it. Does handle shift-hold keys. -- <REF>TEXT:Keycodes
04714	GETTOUCH	(→ #kc T) (→ F) Pops next key from key buffer (if there is one). Does handle shift-hold keys. -- <REF>TEXT:Keycodes
42159	GETKEY	(→ #kc flag) Get a single keypress from the keybuffer, waits if necessary. The key is returned along with TRUE. If an exception happens, returns FALSE. The exception is not handled. Does handle shift-hold keys. -- <REF>TEXT:Keycodes
420A0	GETKEY*	(→ #kc T) (→ F F) (→ {Alrmlist} T F) Get a single keypress from the keybuffer, waits if necessary. The key is returned along with TRUE. If an exception happens (error or alarm), the exceptions is handled and the entry returns FALSE. Does handle shift-hold keys. -- <REF>TEXT:Keycodes
4203C	GetKeyOb	(→ ob) Wait for a single key and return the object associated with this key. Does handle shift-hold keys. -- <REF>TEXT:Keycodes
40454	DoKeyOb	(ob →) Execute ob as if it had been assigned to a key and the key had been pressed.
047C7	REPKEY?	(#kc → flag) Returns TRUE if the key is being pressed. -- <REF>TEXT:Keycodes

40E88	REPEATER	(→) Takes two objects from the runstream, a BINT and a program. The BINT must represent a keycode. The program is evaluated at least once, and then again and again as long as the specified key is being pressed. -- <REF>TEXT:Keycodes
51735	REPEATERCH	(→) Same as REPEATER, but slower, so more appropriate for scrolling and cursor motions. -- <REF>TEXT:Keycodes
42402	KEYINBUFFER?	(→ flag) Returns TRUE if there is at least a key in the key buffer.
41F65	WaitForKey	(→ #kc #flag) Returns next full key press. Does <i>not</i> handle shift-hold keys. -- <REF>TEXT:Keycodes
1A738	Wait/GetKey	(% → ?) Internal WAIT command. Does <i>not</i> handle shift-hold keys. -- <REF>TEXT:Keycodes

4.2.3 The ATTN Flag

42262	ATTN?	(→ flag) Returns TRUE if <u>CANCEL</u> has been pressed.
4243E	?ATTNQUIT	(→) If <u>CANCEL</u> has been pressed, ABORTs program. aka: ?ATTN_QUIT
23768	CKOATTNABORT	(→) Executed by the UserRPL program delimiters x<< and x>> and by xUNTIL. Mainly just ?ATTNQUIT.
4245C	NoAttn?Semi	(→) If <u>CANCEL</u> has been not pressed, drops the rest of the stream.
05040	ATTNFLAG@	(→ #) Recalls <u>CANCEL</u> key counter.
05068	ATTNFLAGCLR	(→) Clears <u>CANCEL</u> key counter. Does not affect the key buffer.

4.2.4 Bad Keys

3FDD1	DoBadKey	(→) Beeps.
3FDC7	DropBadKey	(ob →) Beeps.
3FDBD	2DropBadKey	(ob ob' →) Beeps.

4.2.5 User Keys

41A8D	UserKeys?	(→ flag) Does BINT62 TestSysFlag.
41F3F	GetUserKeys	(→ {}) Returns user keys list (internal format). -- <REF>TEXT:Reserved UserKeys
41C02	(XEQRclKeys)	(→ {}) Recalls all key assignments (in user format) plus status of non defined keys. -- <REF>TEXT:Reserved UserKeys
41B28	(XEQAsnKey)	(ob %rc.p →) Assigns an object to a key, specified in user format.
41E78	(StoUserKeypatch)	(ob #kc #p →) Assigns an object to a key, specified in system format. If ob is NULL-{}, then this actually deletes a key assignment. -- <REF>TEXT:Reserved UserKeys
41F2C	(UserKeys!)	({} →) Stores user keys (list is in internal format). -- <REF>TEXT:Reserved UserKeys
41E32	(StoUserKeys)	({} →) Like <REF>UserKeys!, but also recalculates CRC. -- <REF>TEXT:Reserved UserKeys
41AA1	(Ck&AsnUKeys)	({} →) Stores user keys (list in user format), recalculates CRC.
41B8C	(DelKey)	(#kc #plane →) Deletes that key assignment, recalculates CRC.

41B3C	(XEQDelKeys)	({ } →) Deletes specified keys (in user format).
41B69	(Ck&ClrUKey)	(0 →) (%rc.p →) System version of user word DELKEYS: if 0, deletes all keys, otherwise deletes specified key.
41F52	(PgUserKeys)	(→) Deletes all user keys.
41F13	(ClrUserKeys)	(→) Deletes all user keys and recalculates CRC.
		-- <REF>TEXT:Reserved UserKeys
3FF75	(NonUsrKeyOK?)	(→ flag) Returns TRUE if the keys not defined do their normal actions.
3FF86	(SetNUsrKeyOK)	(→) Keys not defined do their normal actions.
3FF97	(ClrNUsrKeyOK)	(→) Keys not defined just beep when pressed.
3FB1A	Key>StdKeyOb	(#kc #p1 → ob) Recalls the standard assignment of the key. This is the assignment which is active when USER mode is of.
3FA57	Key>U/SKeyOb	(#kc #p1 → ob) If user mode is on, recalls the user object assigned to a key. If user mode is off, recalls the standard assignment instead.
3FA7A	?Key>UKeyOb	
3FF75	(NonUsrKeyOK?)	(→ flag) Returns TRUE if the keys not defined do their normal actions.

4.3 The Menu

4.3.1 Menu Properties

04A41	GETDF	(#menukey → ob) Gets the definition of a menu key from THOUGHTAB. #menukey = #1..#6
04A0B	GETPROC	(#menukey → ob) Gets the definition of a menu key from THOUGHTAB. #menukey = #1..#6. With #7, get the executor.

418F4	LabelDef!	<p>(ob →)</p> <p>Store a program which displays a menu label. Prg has the stack diagram</p> <p>(#col ob →)</p> <p>For example, the LIBS command uses the following program to make all menu label look like directories:</p> <pre>:: DUPNULL\$? ITE MakeStdLabel MakeDirLabel Grob>Menu ;</pre> <p>During execution, INDEX@ will contain the menu key number.</p>
419E4	LastMenuDef!	<p>(menu →)</p> <p>Sets the definition of the last menu. menu is a MenuList or a program, or a Rompointer.</p>
419F4	LastMenuDef@	<p>(→ menu)</p> <p>Recalls the definition of the last menu. menu is a MenuList or a program, or a Rompointer.</p>
4139B	SaveLastMenu	<p>(→)</p> <p>Stores row and definition of current menu as the last menu.</p>
4186E	LastMenuRow!	<p>(#n →)</p> <p>Sets the row of the last menu. #n is not the row, but the index of the first menu key in that row, i.e. 1,7,13,...</p>
41881	LastMenuRow@	<p>(→ #n)</p> <p>Recalls the index to the first menu key in the current row of the last menu. Returns 1 for the first page, 7 for the second page, 13 for the third and so on.</p>
418A4	MenuDef@	<p>(→ menu)</p> <p>Recalls the current menu definition. menu is a MenuList or a program, or a Rompointer.</p>
419C4	MenuExitAct!	<p>(ob →)</p> <p>Store ob as exit action.</p>
3EC85	NoExitAction	<p>(→)</p> <p>Sets NOP as ExitAction. Mostly used to avoid that the menu is saved as the previous menu when a new Menu gets installed.</p>
41848	MenuRow!	<p>(#n →)</p> <p>Sets the menu row. #n is not the row, but the index of the first menu key in that row, i.e. 1,7,13,...</p>
4185B	MenuRow@	<p>(→ #n)</p> <p>Recalls the index of the first menu key in the current menu page. Returns 1 for the first page, 7 for the second page, 13 for the third and so on.</p>

41944	MenuKeyLS!	(ob → ob) Set the action for left-shifted menu keys. The program receives the action part of the menu item as an argument, i.e. {ob-NS ob-LS ob-RS}.
401D4	StdMenuKeyLS	({ob-NS ob-LS ob-RS} → ?) The content of MenuKeyLS for standard menus.
41914	MenuKeyNS!	(og → ob) Set the action for unshifted menu keys. The program receives the action part of the menu item as an argument, i.e. ob-NS or {ob-NS ob-LS ob-RS}.
41924	MenuKeyNS@	(→ ob) Recall the action for unshifted menu keys.
3FCAF	SetKeysNS	(ob →) Sets ob as MenuKeysNS, DoBadKey to LS & RS.
4019D	StdMenuKeyNS	(ob-NS → ?) ({ob-NS ob-LS ob-RS} → ?) The content of MenuKeyNS for standard menus.
41964	MenuKeyRS!	(ob → ob) Set the action for right-shifted menu keys. The program receives the action part of the menu item as an argument, i.e. {ob-NS ob-LS ob-RS}.
417F3	SetRebuild	(→) Sets the flag that the menu needs to be rebuild.
41984	ReviewKey!	(ob →) Store a program which is called with the review key (RS DOWN). The program has the stack diagram (→)
418D4	MenuRowAct!	(ob →) Stores ob as the RowAct menu property.
41741	InitTrack:	(→) Execute the program which is next in the runstream if the directory changes. Used by the VAR menu to set first menurrow when diretory changes, or by the CST menu to rebuild it.

4.3.2 Building Menus

40788	TakeOver	(→) Override the default menu key executer. If this is the first entry in a program, the program can be used in edit mode. When the first in a program in the label slot of a menu key, the program is evaluated to get the label object (most likely a grob).
-------	----------	---

3EC71	NullMenuKey	(→) A placeholder for an empty menu key when defining menu lists.
4085A	Modifier	(→) :: TakeOver ;
407FB	MenuMaker	(→ ob) Quotes next object, and also provides TakeOver. The disassembly is :: TakeOver 'R ; Normally this is used like this: :: MenuMaker menu InitMenu ;
40F86	InitMenu	(menu →) menu is {} or :: settings {} ; Settings override the default settings installed by InitMenu.
40DC0	DoMenuKey	(menu →) :: SetDA12NoCh InitMenu ;
41679	InitMenu%	(%mnu.pg →) (%0 →)
415C9	GetMenu%	(→ %)
41008	StartMenu	(menu #n →) #n is the index of the first menu key on the page, use 1 for the first page, 7 for the second etc. StartMenu does ExitAction (Previous menu!), sets the default menu properties and page. Then it evaluates menu, stores result to MenuKeys and executes SetThisRow.
410C6	SetThisRow	(→) Builds a new TOUCHTAB, SetBadMenu.
41175	LoadTouchTbl	(MenuKey1 .. MenuKeyN #n →) Builds new TOUCHTAB from menukeys.

4.3.3 Menu Display

3866F	SysMenuCheck	(→) Checks menu validity. If DA3NoCh? then nothing. If Track? then ?DoTrackAct@. If Rebuild? then SetThisRow.
3A1CA	?DispMenu	(→) Redisplays the menu now if no key is waiting in the buffer. Even better is this: :: DA3OK?NOTIT ?DispMenu ;
3A1FC	DispMenu.1	(→) Displays menu now.
3A1E8	DispMenu	(→) :: DispMenu.1 SetDAsValid ;

4.3.4 Displaying Menu Labels

3A297	Grob>Menu	(#col grob →) Displays grob as menu label.
3A2B5	Str>Menu	(#col \$ →) Displays string as menu label.
3A2DD	Id>Menu	(#col id →) Displays id as menu label.
3A2C9	Seco>Menu	(#col :: →) Does EVAL then DoLabel.
41904	DoLabel	(#col ob →) If ob is of one of the supported types, displays a menu label. If not, generates a "Bad Argument Type" error.
3A260	(>MENU)	(#col grob →) (#col \$ →) (#col id →) (#col :: →) Works by dispatching the object type.
3A4AB	MakeLabel	(\$ #w #x grob → grob') Inserts \$ into grob using CENTER\$3x5 with y=5.

4.3.5 General Entries

41111	CheckMenuRow	(# → # #')
3A9E7	SetSomeRow	(#n →) with Mod(n,FFFFFh)= 0.
41934	DoMenuKeyNS	(#n →)
40828	MenuKey	(→) Takes NOB from Runstream.
51125	CLEARMENU	(→)
4E266	CHECKMENU	(→)
211B4	(ID_CST)	ID CST
7DEE2	nCustomMenu	(→) Installs the CST menu.
151A6	SolvMenuInit	(→) Sets MenuKeyNS/LS/RS, ReviewKey and LabelDef properties needed by the Solver menu.
3BE54	DoSolvMenu	(→) Installs the solver menu which is also available via 75 MENU.
40350	DoNameKeyLRS	
40337	DoNameKeyRS	

3B211	DoFirstRow	(→) Sets the first row of the current menu.
3A71C	DoNextRow	
3A735	DoPrevRow	

4.4 InputLine and Inputforms

4.4.1 Inputline

42F44	InputLine	(args → \$ T) (args → \$ ob1..obn T) (args → ob1..obn T) (args → F) args = \$pr \$line #pos #I/R #I/A #alph menu #row attn #parse
43200	InputLAttn	
43179	InputLEnter	

4.4.2 Inputform

199EB	DoInputForm	(l1..ln f1..fm #n #m msg \$ → ob1..obn T) (l1..ln f1..fm #n #m msg \$ → F) l = \$ #x #y f = msg #x #y #w #h #type legal dec \$hlp ChDat ChDec res init Starts an input form using the old engine.
0050B0	~IFMenuRow1	(→ { }) Returns the menu for the first menu row of an InputForm.
0060B0	~IFMenuRow2	(→ { }) Returns the menu for the second menu row of an InputForm.

4.5 The Browser Engines

4.5.1 The HP48 Browser Engine

```

0000B3  ~Choose      ( ::Appl $Title ::Convert {} offset → {}' T )
          ( ::Appl $Title ::Convert {} offset → ob T )
          ( ::Appl $Title ::Convert {} offset → F )
          The return value is a list if checkfields are enabled,
          otherwise it is just the selected object. Only FALSE
          is returned when the user presses CANCEL.
          --
          <REF>TEXT:Browser48
0050B3  ~ChooseMenu0 ( → {} )
          Menus with "OK".
          --
          <REF>TEXT:Browser48
0060B3  ~ChooseMenu1 ( → {} )
          Menus with "CANCL", "OK".
          --
          <REF>TEXT:Browser48
0070B3  ~ChooseMenu2 ( → {} )
          Menus with "CHK", "CANCL", "OK".
          --
          <REF>TEXT:Browser48
0660B3  ~ChooseSimple ( $title {items} → ob T )
          ( $title {items} → F )
          Simple interface to the HP48 choose engine. On the
          HP49G, calls ^RunChooseSimple.
          --
          <REF>TEXT:Browser48
0150B3  (~BBMoveTo)  ( # → )
          Moves selection to line and updates display.
          --
          <REF>TEXT:Browser48
0190B3  (~BBRecalOff&Disp)
          ( flag → )
          Recalculates offset of selected item in page, and re-
          draws lines if the flag is TRUE.
          --
          <REF>TEXT:Browser48
0220B3  (~BBRunEntryProc)
          ( → )
          Sends message 85 to ::Appl, thus running the user-
          defined start-up procedure.
          --
          <REF>TEXT:Browser48
0230B3  (~BBReReadPageSize)
          ( → )
          Re-reads the size of the page (message 57).
          --
          <REF>TEXT:Browser48
0240B3  (~BBReReadHeight)
          ( → )
          Re-reads the height of the browser line (message 58).
          --
          <REF>TEXT:Browser48

```

0250B3	(~BBReReadCoords)	(→) Re-reads the coordinates of the browser box (message 63). -- <REF>TEXT:Browser48
0260B3	(~BBReReadWidth)	(→) Re-reads the width of the browser line (message 59). -- <REF>TEXT:Browser48
0280B3	(~BBRunENTERAction)	(→) Sends message 96 to ::Appl, thus running the OK action. It does not check the value returned and never exits. -- <REF>TEXT:Browser48
0290B3	(~BBRunCanclAction)	(→) Sends message 91 to ::Appl, thus running the <u>CANCEL</u> action. It does not check the value returned and never exits. -- <REF>TEXT:Browser48
02F0B3	(~BBReDrawBackgr)	(→) Redraws the background. -- <REF>TEXT:Browser48
0370B3	(~BBGetNGrob)	(#n → grob) Returns nth element as a grob. -- <REF>TEXT:Browser48
0380B3	(~BBGetNStr)	(#n → \$) Returns nth element as a string. -- <REF>TEXT:Browser48
03B0B3	(~BBRereadChkEnbl)	(→) Re-reads whether checkmarks are enabled. (Message 61). -- <REF>TEXT:Browser48
03C0B3	(~BBRereadFullScr)	(→) Re-reads whether to use full-screen mode. (Message 60). -- <REF>TEXT:Browser48
03D0B3	(~BReReadMenus)	(→) Re-reads the menu. (Message 83). -- <REF>TEXT:Browser48

```

03E0B3      (~BBReReadNElems)      ( → )
Re-reads the number of elements. (Message 62).
--
<REF>TEXT:Browser48
03F0B3      (~BBGetN)              ( #n → ob )
Returns nth element.
--
<REF>TEXT:Browser48
04B0B3      (~BBIIsChecked?)      ( #n → flag )
Returns whether the given element is checked.
--
<REF>TEXT:Browser48
0520B3      (~BBUpArrow)          ( → grob )
Returns up arrow as grob
--
<REF>TEXT:Browser48
0530B3      (~BBDownArrow)        ( → grob )
Returns down arrow as grob
--
<REF>TEXT:Browser48
0540B3      (~BBSpace)           ( → grob )
Returns a space as grob.
--
<REF>TEXT:Browser48
0590B3      (~BBPgDown)          ( → )
Go down one page.
--
<REF>TEXT:Browser48
05A0B3      (~BBPgUp)           ( → )
Go up one page.
--
<REF>TEXT:Browser48
05B0B3      (~BBEmpty?)         ( → flag )
Returns TRUE if the browser has no elements.
--
<REF>TEXT:Browser48
05C0B3      (~BBGetDefltHeight)  ( → # )
Returns height of lines based on the font that will be
used. This value is the default height of the browser.
Equivalent to FPTR 2 64.
--
<REF>TEXT:Browser48
0100E0      ~BRbrowse
0120E0      ~BRoutput
0180E0      ~BRRclC1             ( → )
:: LAM 'BR5 ;
0170E0      ~BRRclCurRow
:: LAM 'BR3 ;

```

```
0030E0    ~BRStoC1
                                :: ' LAM 'BR5 STO ;
```

4.5.2 The HP49 Browser Engine

4.6 The Parametrized Outer Loop (POL)

38985	ParOuterLoop	(Disp Keys NonAppKeys? DoStdKeys? menu #row suspendOK? ExitCond AppErr →)
389BC	POLSaveUI	(Disp Keys NonAppKeys? DoStdKeys? menu #row suspendOK? ExitCond AppErr →)
38A64	POLSetUI	Saves current UI to LAMSavedUI. <see>ParOuterLoop
38AEB	POLKeyUI	Sets new UI, same arguments as to ParOuterLoop. (→)
38B09	(1POLKeyUI)	Displays, reads and evaluates keys according to set UI. (→)
38B45	(POLKeyErr)	Executes UI once, doesn't check exit condition. (→)
38994	(POLSet&KeyUI)	Handles error caused by a key press. <see>ParOuterLoop :: POLSetUI POLKeyUI ;
38B90	POLRestoreUI	(→)
38B77	POLResUI&Err	Restores saved UI from LAMSavedUI. (→)
389CB	(Rc1UI)	Restores saved UI and executes ERRJMP.
38A11	(Rc1HPUI)	Recalls current ParOuterLoop UI
38BD6	(NormAppFlags)	Recalls system UI.
38A3E	(SavedUIILS)	Sets default application flags.
38A3E	(LAMSavedUI)	{ LAMSavedUI }
38C08	AppDisplay!	(ob →)
38C18	AppDisplay@	(→)
38C28	(NoAppDisplay!)	
38C38	AppKeys!	(ob →)
38C58	AppKeys0	???
38C48	(AppKeys@)	
38C58	(NoAppKeys!)	

38C68	AppExitCond!	(ob →)
38C78	AppExitCond@	(→ ob)
38C88	(NoAppExitCnd!)	
38C98	AppError!	(ob →)
38CAB	AppError@	(→ ob)
38CBE	(NoAppError!)	(→)
38CD1	(AppError?)	(→ flag)
38CDF	(SetAppError)	(→)
38CED	(ClrAppError)	(→)
38CFB	AppMode?	(→ flag)
		Is currently a POL active?
38D09	SetAppMode	(→)
38D17	ClrAppMode	(→)
38D25	(NAppKeyOK?)	(→ flag)
38D33	SetNAppKeyOK	(→)
38D41	(ClrNAppKeyOK)	(→)
38D4F	(DoStdKeys?)	(→ flag)
38D5D	SetDoStdKeys	(→)
38D6B	(ClrDoStdKeys)	(→)
1446F	SuspendOK?	(→ flag)
		Does the current user interface allow suspension?
14483	nohalt	(→ ob)
		:: LAM 'nohalt ;
38D79	(AppSuspOK?)	(→)
38D8A	SetAppSuspOK	(→)
38D9B	ClrAppSuspOK	(→)
38BD6	(InitPOLVars)	

4.7 Editor Commands

4.7.1 Status

53A4A	EditLExists?	(→ flag)
		Does an EditLine exist?
4488A	NoEditLine?	(→ flag)
		Does no EditLine exist?
44683	EDITLINE\$	(→ \$)
		Returns a copy of the current command line to the stack. Same as RCL_CMD.
13EF1	(CURSOR@)	(→ #)
		Recalls the current cursor position.

444A5 `CURSOR_END?` (\rightarrow flag)
 Checks if the cursor is at the end of a line or at the end of the file. Works by checking the current character against newline and `CHR_00`.

4.7.2 Display Window

11432 (FIRSTC@) (\rightarrow #)
 Column of the left display window edge.

11462 (FIRSTC+) (\rightarrow)
 Increases the position of the left window edge by one.

11442 (SETFIRSTC_0) (\rightarrow)
 Sets the position of the left display window edge to zero.

13E85 `CURSOR_OFF` (\rightarrow #)
 Cursor column relative to left edge of display window.

13ED2 (CURSOR_OFF+) (\rightarrow)
 Increases the `CURSOR` offset by one.

13EBC (CURSOR_OFF0) (\rightarrow)
 Sets the cursor offset to zero.

4.7.3 Inserting Text

42CFB (InsertEcho) (\$ \rightarrow)
 Inserts string at current cursor position in `EditLine`.

42BD4 `Echo$Key` (\$/chr \rightarrow)
 Same as `CMD_PLUS`.

42AE4 `EchoChrKey` (\$/chr \rightarrow)
 Same as `CMD_PLUS`, but first `?TogU/LCase`.

42BB6 `Echo$NoChr00` (\$ \rightarrow)
 Inserts string at current cursor position in `EditLine`.

40DD4 `DoDelim` (\rightarrow)
 Takes a character or string from the runstream and inserts it.

40DF7 `DoDelims` (\rightarrow)
 Takes a character or a string from the runstream, inserts it and moves the cursor back by one character.

53A2E `INSERT_MODE` (\rightarrow)
 Turns insert mode on. In insert mode, new characters do not overwrite old ones.

448C1	?TogU/LCase	(chr → chr') Toggle upper/lowercase of character if some condition is fulfilled.
53A3C	INSERT?	(→ flag) Returns TRUE if insert mode is active.

4.7.4 Moving the Cursor

4325A	SetCursor	(# →) ({# #'} →) Sets the cursor to the given position. For the list argument, the numbers are row and column.
13F29	(SETCURSOR)	

4.7.5 Starting the Editor

42D46	ViewLevel1	(ob → ob') Edits the object in level 1.
444EE	Char>Edit	
42D82	CharEdit	
53A90	ClrNewEditL	
42D32	EditLevel1	(ob → ob')
42DC8	ObEdit	(ob → ob' T) (ob → F) Edits object. When the user cancels, only FALSE is returned. Otherwise the changed object along with TRUE is returned.

443CB	EditString	<p>(\$ →)</p> <p>Starts editing the string in the command line when the current program exits. This is the entry to use if a program should exit with the command line. Use <code>InitEdLine</code> before this entry to clear the command line (if desired) - if not, the string is inserted into the existing command line. All code after this entry will be executed <i>before</i> control is handed to the editor application. For example:</p> <pre> :: "SOME STRING" DUPLen\$ SWAP (get length) InitEdLine (clear the editline) EditString (string to editline) STO_CURS_POS2 (cursor at end) "Starting editor..." FlashMsg (display *before* edit) ; </pre> <p>Note that when you press ENTER after editing, the command line will be parsed normally.</p>
42E86	Rc1&Do:	<p>(id →)</p> <p>Executes the program which is next in the runstream on the contents of the variable. The program typically is an edit command, with the stack diagrams</p> <pre> (ob → ob' T) (ob → F) </pre> <p>If the flag is TRUE, ob' is stored back into the original variable.</p>
42E27	Roll&Do:	<p>(# →)</p> <p>Does ROLL and then executes the program which is next on the runsteam. So the program is applied to the object on level #. Typically, this is an edit command, with the stack diagram</p> <pre> (ob → ob) </pre> <p>After the program exits, UNROLL is used to put the object back to the right stack position. This entry is probably used in the interactive stack.</p>

4.7.6 Miscellaneous

3BDFa	EditMenu	<p>(→ { })</p> <p>Returns the Editor menu.</p>
3E3E1	<DelKey	<p>(→ { })</p> <p>Returns the <code>←DEL</code> menu key.</p>
3E4CA	>DelKey	<p>(→ { })</p> <p>Returns the <code>DEL→</code> menu key.</p>

3E2DD	<SkipKey	(→ {}) Returns the $\overleftarrow{\text{SKIP}}$ menu key.
3E35F	>SkipKey	(→ {}) Returns the $\overrightarrow{\text{SKIP}}$ menu key.
44277	InitEd&Modes	(→) :: InitEdLine InitEdModes ;
4428B	InitEdLine	(→) :: DEL_CMD ;
44394	InitEdModes	(→)
40C76	SaveLastEdit	(\$ →) Calls CMD_STO if history is on.
40C94	CMDSTO	(\$ →) Adds string to the list of the last 4 commands, accessible with the $\overline{\text{CMD}}$ key.

4.8 Entries Related to the Matrix Editor and Matrix Operations

44C31	DoNewMatrix	(→ []/[[]]) Start matrix editor to enter a new matrix.
44FE7	DoOldMatrix	([] → []') Edit an existing matrix.

4.9 The Display

4.9.1 Display Organization

1314D	TOADISP	(→) Sets the text display as the active.
13135	TOGDISP	(→) Sets the graphic display as the active.
13167	(GDISPON?)	(→ flag) Returns a flag indicating whether the graphic display is active.
12655	ABUFF	(→ textgrob) Returns the text grob to the stack.
12665	GBUFF	(→ graphgrob) Returns the graphic grob to the stack. The HP49 extable address for ExitAction! is the same, but this must be a bug.
12635	HARDBUFF	(→ dispgrob) Returns the current grob to the stack.

12645	HARDBUFF2	(→ menugrob) Returns the menu grob to the stack.
0E128	HBUF_X_Y	(→ HBgrob #x #y)
12B6C	HARDHEIGHT	(→ #height) Returns the height of HARDBUFF.
12B58	(HBUFFDIMw)	(→ #width)
5187F	GBUFFGROBDIM	(→ #height #width) Returns dimensions of graphic grob.

4.9.2 Preparing the Display

130AC	RECLAIMDISP	(→) Activates the text grob, clears it and sets the default size.
39531	ClrDA1IsStat	(→) Suspends clock display.
4E360	MENUOFF?	(→ flag) Returns TRUE if the menu grob is off.
4E2CF	TURNMENUOFF	(→) Turns off menu display, enlarges ABUFF to fill screen.
4E347	TURNMENUON	(→) Turns menu grob on.
4E2AC	MENUOFF	(→)
130CA	(RSZVDISP)	(→) Sets standard size for currently displayed grob.
1297D	(BROADENHBUFF)	(#cols →) Broadens currently displayed grob.
12964	(HEIGHTENHBUFF)	(#rows →) Heightens currently displayed grob.
12BB7	(BROADENGROB)	(grob #cols →) Broadens graph or text grob.
12DD1	HEIGHTENGROB	(grob #rows →) Heightens graph or text grob.
13043	(KILLADISP)	(→) Clears text display.
13061	KILLGDISP	(→) Clears graph display by setting it to NULLGROB. See DOERASE.
4B60C	DOERASE	(→) Erases the graphics display grob without changing its size.

4.9.3 Immediate Refresh

386A1	SysDisplay	(→) Redisplays all required areas. Does it immediately, without waiting for the current command to finish.
3A00D	DispEditLine	(→) Just calls <code>DispCommandLine</code> .
3A1CA	?DispMenu	(→) Redisplays the menu now if no key is waiting in the buffer. Even better is this: <code>:: DA30K?NOTIT ?DispMenu ;</code>
3A1FC	DispMenu.1	(→) Displays menu now.
3A1E8	DispMenu	(→) <code>:: DispMenu.1 SetDAsValid ;</code>
39B85	?DispStack	(→) Redisplays the stack now if necessary.
3959C	?DispStatus	(→) Redisplays the status area now if necessary.
395BA	DispStatus	(→) Displays the status area now.
39B0A	DispStsBound	(→) Displays a horizontal line at y=14, normally the separation between header and stack.
39515	(DispTime?)	
39AF1	DispTimeReq?	(→ flag) Is time display required? Checks system flag 40 and something else.
398F4	DispDir?Tim1	
39958	DispDir?Tim2	
3988B	DispDir?Time	
430CF	DispILPrompt	(→) Redisplays the <code>InputLine</code> prompt, i.e. refreshes the region between the command line and the header during <code>InputLine</code> . Requires a string (the prompt) in 4LAM.

4.9.4 Controlling Display Refresh

390CC	ClrDA10K	(→)
390E5	ClrDA2aOK	(→)
390FE	ClrDA2bOK	(→)
39117	ClrDA20K	(→)

3912B	ClrDA30K	(→)
39144	ClrDAsOK	(→)
38DAC	DA10K?	(→ flag)
38DFC	(DA2aOK?)	(→ flag)
38E4C	(DA2bOK?)	(→ flag)
38E9C	(DA20K?)	(→ flag)
38EB5	DA30K?	(→ flag)
38F05	(DAsOK?)	(→ flag)
38FB9	DA2aLess10K?	(→ flag)
38F28	DA10K?NOTIT	(→) Does DA10K?, NOT then IT.
38F41	DA2aOK?NOTIT	(→) DA2aOK?, NOT then IT.
38F5A	DA2bOK?NOTIT	(→) DA2bOK?, NOT then IT.
38F73	DA30K?NOTIT	(→) Does DA30K?, NOT then IT.
3902C	SetDA1Temp	(→)
39045	SetDA2aTemp	(→)
39059	SetDA2bTemp	(→)
3938D	(ClrDA2bTemp)	(→)
39207	SetDA20KTemp	(→)
39072	SetDA3Temp	(→)
3921B	SetDA12Temp	(→)
3922F	SetDAsTemp	(→)
3932B	(SetDA1TempF)	(→)
39339	(ClrDA1TempF)	(→)
3931D	(DA1TempF)	(→)
39355	(SetDA2aTempF)	(→)
39363	(ClrDA2aTempF)	(→)
39347	(DA2aTempF?)	(→ flag)
3937F	(SetDA2bTempF)	(→)
3938D	(ClrDA2bTempF)	(→)
39371	(DA2bTempF?)	(→ flag)
393A9	(SetDA3TempF)	(→)
393B7	(ClrDA3TempF)	(→)
3939B	(DA3TempF?)	(→ flag)
38FD2	SetDA1Valid	(→)
38FEB	SetDA2aValid	(→)

38FFF	SetDA2bValid	(→)
3915D	SetDA2Valid	(→)
39018	SetDA3Valid	(→)
39171	(SetDAsValid)	(→)
39283	(SetDA1ValidF)	(→)
39291	(ClrDA1ValidF)	(→)
39275	(DA1ValidF?)	(→ flag)
392AD	(SetDA2aValidF)	(→)
392BB	(ClrDA2aValidF)	(→)
3929F	(DA2aValidF?)	(→ flag)
392D7	(SetDA2bValidF)	(→)
392E5	(ClrDA2bValidF)	(→)
392C9	(DA2bValidF?)	(→ flag)
39301	SetDA3ValidF	(→)
3930F	(ClrDA3ValidF)	(→)
392F3	(DA3ValidF?)	(→ flag)
39248	(DAsBad?)	(→ flag)
		Is any DA "Bad"?
3947B	SetDA1Bad	(→)
38DE8	(SetDA1BadT)	(→ T)
		(SetDA1Bad TRUE)
39489	ClrDA1Bad	(→)
3946D	(DA1Bad?)	(→ flag)
394A5	SetDA2aBad	(→)
38E38	(SetDA2aBadT)	(→ T)
		(SetDA2aBad TRUE)
394B3	ClrDA2aBad	(→)
39497	DA2aBad?	(→ flag)
394CF	SetDA2bBad	(→)
38E88	(SetDA2bBadT)	(→ T)
		(SetDA2bBad TRUE)
394DD	ClrDA2bBad	(→)
394C1	(DA2bBad?)	(→ flag)
394F9	SetDA3Bad	(→)
38EF1	(SetDA3BadT)	(→ T)
		(SetDA3Bad TRUE)
39507	ClrDA3Bad	(→)
394EB	(DA3Bad?)	(→ flag)
393D3	SetDA1NoCh	(→)


```

393E1      (ClrDA1NoCh)          ( → )
393C5      (DA1NoCh?)        ( → flag )
393FD      SetDA2aNoCh      ( → )
3940B      (ClrDA2NoCh)    ( → )
393EF      (DA2aNoCh?)    ( → flag )
39427      SetDA2bNoCh     ( → )
39435      (ClrDA2bNoCh)   ( → )
39419      DA2bNoCh?      ( → flag )
3918A      SetDA2NoCh      ( → )
3919E      SetDA12NoCh     ( → )
39451      SetDA3NoCh      ( → )
3945F      (ClrDA3NoCh)    ( → )
39443      (DA3NoCh?)     ( → flag )
391C6      SetDA13NoCh     ( → )
391B2      SetDA23NoCh     ( → )
391DA      (SetDA12a3NCh)  ( → )
                                     aka: SetDA12a3NoCh
391EE      SetDA123NoCh    ( → )
391EE      (SetDAsNoCh)    ( → )
39086      SetDA2Echo      ( → )
39086      SetDA2aEcho     ( → )
39523      SetDA1IsStat    ( → )
3957A      SetNoRollDA2    ( → )
3958B      ClrNoRollDA2    ( → )
39569      (NoRollDA2?)   ( → flag )
39086      (?SetEditRoll)  :: EditExst?NOT ITE SetDA2RollF SetDA2aNoCh
                                     ;
39515      (DA1IsStat?)    ( → flag )
39515      (DA1IsStatus?)  ( → flag )
3954D      (SetDA2bIsEdL)  ( → )
3953F      (DA2bIsEdL?)   ( → flag )
3955B      (ClrDA2bIsEdL)  ( → )
3954D      (SetDA2bEdit)   ( → )
3955B      (ClrDA2bEdit)   ( → )
3953F      (DA2bEdit?)    ( → flag )
390A4      MENoP&FixDA1
390B3      MENP&FixDA12
38F8C      (InitDispModes) ( → )

```

142FB Ck&Freeze (% →)
Internal FREEZE.

4.9.5 Clearing the Display

126DF BLANKIT (#startrow #rows →)
Clears #rows from HARDBUFF, starting at #startrow.

134AE CLEARVDISP (→)
Clears HARDBUFF.

0E083 Clr8 (→)
Clears top eight rows (first status line).

0E097 Clr8-15 (→)
Clears 2nd status line.

0E06F Clr16 (→)
Clears top 16 rows.

3A546 BlankDA1 (→)
Clears status area from HARDBUFF.

3A591 BlankDA2a (→)
Clears display area DA2a.

3A55F BlankDA2 (→)
Clears display areas DA2a and DA2b.

3A578 BlankDA12 (→)
Clears display areas DA1 and DA2

01F6D CLCD10 (→)
Clears status and stack areas.

01FA7 CLEARLCD (→)
Clears whole display.

5046A DOCLLCD (→)
Like user word <REF>CLLCD.

4.9.6 Annunciator and Modes Control

11361 SetLeftAnn (→)
Sets left-shift annunciator.

1136E ClrLeftAnn (→)
Clears left-shift annunciator.

11347 SetRightAnn (→)
Sets right-shift annunciator.

11354 ClrRightAnn (→)
Clears right-shift annunciator.

1132D SetAlphaAnn (→)
Sets alpha annunciator.

1133A ClrAlphaAnn (→)
Clears alpha annunciator.

11543	(SetLock)	(→) Sets alpha mode.
1156C	(ClrLock)	(→) Clears alpha mode.
40D25	LockAlpha	(→) Sets alpha mode, annunciators, etc.
40D39	UnLockAlpha	(→) Clears alpha mode, annunciators, etc.
11501	(Lock?)	(→ flag) Is alpha mode set?
11320	(ClrBusyAnn)	(→) Clears the busy annunciator.
11533	SetPrgmEntry	(→) Sets program-entry mode.
1155C	(ClrPrgmEntry)	(→) Clears program-entry mode.
11511	PrgmEntry?	(→ flag) Is program-entry mode set?
3EDF2	(Do1st/1st+:)	(→ :: ob1 ; (PRG mode)) (→ :: ob1 <rest> (no PRG mode)) If in program mode, executes only the next object after it. If not, execution continues normally.
3EE1A	Do1st/2nd+:	(→ :: <ob1> ; (PRG mode)) (→ :: <ob2> <rest> ; (no PRG mode)) If in program mode, executes the next object after it. If not in program mode, executes the rest of the stream starting at the second object after it.
53976	SetAlgEntry	(→) Sets algebraic-entry mode.
53984	ClrAlgEntry	(→) Clears algebraic-entry mode.
53968	AlgEntry?	(→ flag) Is algebraic-entry mode set?
408AA	ImmedEntry?	(→ flag) Returns TRUE if immediate-entry mode (program and algebraic-entry modes cleared).
40E3D	?ClrAlg	(→) Clears AlgEntry mode if set.
40E5B	?ClrAlgSetPr	(→) Clears AlgEntry mode if set and sets ProgramEntry mode.

4.9.7 Window Coordinates

0E0D3	TOP8	(→ HBgrob #x1 #y #x1+131 #y1+8) Returns coordinates of first status line.
0E0FB	Rows8-15	(→ HBgrob #x1 #y1+8 #x1+131 #y1+16) Returns coordinates of second status line.
0E0AB	TOP16	(→ HBgrob #x1 #y1 #x1+131 #y1+16) Returns coordinates of status area.
137B6	WINDOWCORNER	(→ #y #x) Gets coordinates of corner of window. Note the order of #x and #y.
0E128	HBUFF_X_Y	(→ HBgrob #x #y) Returns current grob and window coordinates.
515FA	LEFTCOL	(→ #x) Gets x-coordinate of left column.
5160E	RIGHTCOL	(→ #x) Gets x-coordinate of right column.
515A0	TOPROW	(→ #y) Gets y-coordinate of top row.
515B4	BOTROW	(→ #y) Gets y-coordinate of bottom row.
13679	WINDOWXY	(#y #x →) Sets corner coordinates. The name really should be WINDOWYX
13695	(REDISPHBUFF)	(→) Sets #0 and #0 as window corner coordinates.

4.9.8 Scrolling the Display

131C8	WINDOWUP	(→) Moves display one pixel up.
13220	WINDOWDOWN	(→) Moves display one pixel down.
134E4	WINDOWLEFT	(→) Moves display one pixel left.
1357F	WINDOWRIGHT	(→) Moves display one pixel right.
4D132	SCROLLUP	(→) Moves display one pixel up, checks for corresponding key being pressed.
4D16E	SCROLLEDOWN	(→) Moves display one pixel down, checks for corresponding key being pressed.
4D150	SCROLLLEFT	(→) Moves display one pixel left, checks for corresponding key being pressed.

4D18C	SCROLLRIGHT	(→) Moves display one pixel right, checks for corresponding key being pressed.
51690	JUMPTOP	(→) Jumps to top of display.
516AE	JUMPBOT	(→) Jumps to bottom of display.
516E5	JUMPLEFT	(→) Jumps to left of display.
51703	JUMPRIGHT	(→) Jumps to right of display.
5162C	WINDOWTOP?	(→ flag) Is window at the top?
51645	WINDOWBOT?	(→ flag) Is window at the bottom?
5165E	WINDOWLEFT?	(→ flag) Is window at the left?
51677	WINDOWRIGHT?	(→ flag) Is window at the right?
12996	(ScreenUpN)	((#n →) moves stk displ up #n lines) (#n) Moves stack display up #n lines.
12A4A	(ScreenDnN)	((#n →) moves stk displ down #n lines) (#n) Moves stack display down #n lines.
12A0D	(ScreenUp)	moves stk displ up 1 line Moves stack display up one line.
12AF6	(ScreenDn)	moves stk displ down 1 line Moves stack display down one line.

4.9.9 Displaying Objects

14C17	sstDISP	(ob →) Displays ob in status line. Used for single stepping during debugging.
4F052	WINDOW#	(#x #y →) Internal PVIEW, displays PICT starting at the given coordinates.

4.9.10 Displaying Text

140AB	DODISP	(ob %row →) Displays any object in specified row.
1245B	DISPROW1	(\$ →) aka: DISP@01, BIGDISPROW1
12725	DISPROW1*	(\$ →) Displays relative to window corner.

0E029	DISPROW1*!	(\$ →) Does Clr8 then DISPROW1*.
1246B	DISPROW2	(\$ →) aka: DISP@09, BIGDISPROW2
12748	DISPROW2*	(\$ →) Displays relative to window corner.
1247B	DISPROW3	(\$ →) aka: DISP@17, BIGDISPROW3
1248B	DISPROW4	(\$ →) aka: DISP@25, BIGDISPROW4
1249B	DISPROW5	(\$ →)
124AB	DISPROW6	(\$ →)
124BB	DISPROW7	(\$ →)
124CB	DISPROW8	(\$ →) May not be possible depending on the size of the font and whether the menu is on or off.
12429	DISPN	(\$ #row →) aka: BIGDISPN
3A4CE	Disp5x7	(\$ #start #max →) Displays string on multiple lines, starting at #start and no using more than #max rows. New lines must be manually specified. Segments longer than 22 characters are truncated and appended with "...".
39632	Blank&GROB!	(\$ #x #x1 #x2 →) Clears HARDBUFF between (#x1, 0) and (#x2, 6). Converts string to grob with small characters and displays it at (#x, 0).
0E047	(Save16)	(→ grob) Returns top 16 rows.
0E05B	(Restore16)	(grob →) Restores top 16 rows.
1270C	DISPSTATUS2	(\$ →) Displays message in status area using two lines.
4E6EF	DispCoord1	(\$ →) Displays \$ in menu grob using minifont.
4A055	DISPCOORD2	(\$ →) Displays \$ in menu grob using minifont and waits for a key. Then refreshes menu display.

4.9.11 Messages and Boxes

12B85	FlashMsg	(\$ →) Displays message in status area, then restores it to normal.
-------	----------	--

38926	FlashWarning	(\$ →) Displays message in a message box and beeps. Waits for OK to be pressed.
38908	DoWarning	(\$ →) Displays message, beeps and freezes status area.
0000B1	~DoMsgBox	(\$ #x #y grob menu → T) Displays a message box with a grob in the upper left corner and the specified menu. If no grob is desired, use MINUSONE. The meaning of #x and #y is unclear - it seems that any BINT will do.
0020B1	~MsgBoxMenu	(→ { }) The message box menu, with just the OK key.

4.10 Graphics

4.10.1 Built-in Grobs

505B2	(NULLPAINT)	(→ grob) 0x0 Null grob
13D8C	CURSOR1	(→ grob) 6x10 Insert cursor (arrow)
13DB4	CURSOR2	(→ grob) 6x10 Replace cursor (solid box)
66EF1	SmallCursor	(→ grob)
66ECD	MediumCursor	(→ grob)
66EA5	BigCursor	(→ grob)
5053C	CROSSGROB	(→ grob) 5x5 Cross cursor ("+")
5055A	MARKGROB	(→ grob) 5x5 Mark symbol ("x")
39B2D	(LineGrob)	131x2 line 131x2 Line (status area divider)
3A337	(NullMenuLbl)	21x8 normal menu key
3A399	(BoxLabelGrob)	21x8 menu key with box
3A3FB	(DirLabelGrob)	21x8 directory menu key
3A45D	(InvLabelGrob)	21x8 inverse menu key
0850B0	~grobAlertIcon	9x9 Alert grob
0860B0	~grobCheckKey	21x8 Check Key menu grob A tickmark and "CHK" in a menu grob.

4.10.2 Dimensions

50578	GROBDIM	(grob → #height #width)
5179E	DUPGROBDIM	(grob → grob #height #width)
63C04	GROBDIMw	(grob → #width)
4F7E6	CKGROBFITS	(g1 g2 #n #m → g1 g2' #n #m) Shrinks g2 if it does not fit in g1.
511E3	CHECKHEIGHT	(grob #height →) Forces grob (ABUFF/GBUFF) to be at least 64 rows high.

4.10.3 Grob Handling

11679	GROB!	(grob1 grob2 #x #y →) Stores grob1 into grob2. Bang type.
4F78C	GROB+#	(flag grob1 grob2 #x #y → grob') Inserts grob2 into the specified position of grob1, using OR (if flag is TRUE) or XOR (if flag is FALSE). Does all necessary checks first.
11A6D	GROB!ZERO	(grob #x1 #y1 #x2 #y2 → grob') Blanks a rectangular region of the grob. Bang type.
6389E	GROB!ZERODRP	(grob #x1 #y1 #x2 #y2 →) Blanks a rectangular region of the grob. Probably only useful if grob is the text or graphics grob (see section on display-organization). Bang type.
1192F	SUBGROB	(grob #x1 #y1 #x2 #y2 → grob') Returns specified portion of grob.
128B0	XYGROBDISP	(#x #y grob →) Stores grob in HARDBUFF with upper left corner at (#x,#y). HARDBUFF is expanded if necessary.
12F94	GROB>GDISP	(grob →) Stores new graph grob.
1158F	MAKEGROB	(#height #width → grob) Creates a blank grob.
4B323	MAKEPICT#	(#w #h →) Creates blank graph grob. Minimum size is 131x64. Smaller grobs will be automatically resized.
122FF	INVGROB	(grob → grob') Inverts grob data bits. Bang type.
1384A	PIXON	(#x #y →) Sets pixel in text grob.
1383B	PIXOFF	(#x #y →) Clears pixel in text grob.
13992	PIXON?	(#x #y → flag) Is pixel in text grob on?

13825	PIXON3	(#x #y →) Sets pixel in graph grob.
1380F	PIXOFF3	(#x #y →) Clears pixel in graph grob.
13986	PIXON?3	(#x #y → flag) Is pixel in graph grob on?
51893	ORDERXY#	(#x1 #y1 #x2 #y2 → #x1' #y1' #x2' #y2') Orders the bints to be appropriate for defining a rectangle in a grob. Swaps #x1 and #x2 if #x2<#x1. Swaps #y1 and #y2 if #y2<#y1.
518CA	ORDERXY%	(%x1 %y1 %x2 %y2 → %x1' %y1' %x2' %y2') ORDERXY# with real numbers.
50B17	LINEON	(#x1 #y1 #x2 #y2 →) Draws a line in text grob.
50B08	LINEOFF	(#x1 #y1 #x2 #y2 →) Clears a line in text grob.
50AF9	TOGLINE	(#x1 #y1 #x2 #y2 →) Toggles a line in text grob.
50AEA	LINEON3	(#x1 #y1 #x2 #y2 →) Draws a line in graph grob.
50758	DRAWLINE#3	(#x1 #y1 #x2 #y2 →) Draws a line in graph grob. x1<x2 is not required.
50ACC	LINEOFF3	(#x1 #y1 #x2 #y2 →) Clears a line in graph grob.
50ADB	TOGLINE3	(#x1 #y1 #x2 #y2 →) Toggles a line in graph grob.
5072B	TOGGLELINE#3	(#x1 #y1 #x2 #y2 →) Toggles line in graph grob. x1<x2 is not required.
4E582	DRAWBOX#	(#x1 #y1 #x2 #y2 →) Draws rectangle in graph grob.
503D4	DOLCD>	(→ grob) Returns current display.
50438	DO>LCD	(grob →) Grob to display.
659DE	Symb>HBuff	(symb →) Displays symbolic in HARDBUFF in Equation Writer form. Enlarges HARDBUFF if necessary, so use RECLAIMDISP after.

4.10.4 Creating Menu Label Grobs

3A328	MakeStdLabel	(\$ → grob) Makes standard menu label.
3A38A	MakeBoxLabel	(\$ → grob) Makes label with a box.

3A3EC	MakeDirLabel	(\$ → grob) Makes directory label.
3ED6B	(DirLabel:)	(→ grob) Makes directory label with next string in the stream. Usage: :: DirLabel: \$;
3A44E	MakeInvLabel	(\$ → grob) Makes inverse label.
3EC99	Box/StdLabel	(\$ flag → grob) If TRUE makes box label, otherwise makes standard label.
3ED0C	Std/BoxLabel	(\$ flag → grob) If TRUE makes standard label, otherwise makes box label.
3ECB2	Box/StdLbl:	(→ grob) Does Box/StdLabel with the next two objects from the stream. Usage: :: Box/StdLbl: \$ <test> ;
3ECEE	(FBox/StdLbl:)	(→ grob) Takes a string and a bint from the runstream. Tests the system flag specified, does Box/StdLabel. Usage: :: FBox/StdLbl: \$ #flag ;
3ED25	(BBox/StdLbl:)	(→ grob) Takes a string and a bint from the runstream. Does BASE then EQ, and finally Box/StdLabel. Usage: :: BBox/StdLabel: \$ #base ;
3ED48	(MBox/StdLbl:)	(→ grob) Takes a string and a bint from the runstream. Does NumbMode and EQ, then Box/StdLabel. Usage: :: MBox/StdLbl: \$ #mode ;
3ECD0	(FStd/BoxLbl:)	(→ grob) Takes a string and a bint from the runstream. Tests the system flag specified, does Std/BoxLabel. Usage: :: FStd/BoxLbl: \$ #flag ;
3A297	Grob>Menu	(#col grob →) Displays grob as menu label.
3A2B5	Str>Menu	(#col \$ →) Displays string as menu label.
3A2DD	Id>Menu	(#col id →) Displays id as menu label.
3A2C9	Seco>Menu	(#col :: →) Does EVAL then DoLabel.
41904	DoLabel	(#col ob →) If ob is of one of the supported types, displays a menu label. If not, generates a "Bad Argument Type" error.

3A260	(StdLabelDef)	(#col grob →) (#col \$ →) (#col id →) (#col :: →) Works by dispatching the object type.
-------	---------------	---

4.10.5 Converting Strings to Grobs

11CF3	\$>BIGGROB	(\$ → grob) Makes grob of the string using the large font (5x9).
11D00	\$>GROB	(\$ → grob) Makes grob of the string using the system font. Linefeed does <i>not</i> make new line.
11F80	\$>grob	(\$ → grob) Makes grob of the string using the minifont. Linefeed does <i>not</i> make new line.
05F0B3	(~\$>grobOrGROB)	(\$ → grob) Converts string to a grob using either the current font or the minifont, depending on system flag 90.
1200C	RIGHT\$3x6	(\$ #n → flag grob) Transforms string into grob (using the minifont), then takes all characters starting after column #n. flag is FALSE if #n is greater than the width of the grob. In this case, the whole grob is returned.
1215E	CENTER\$3x5	(grob #x #y \$ #w → grob') Creates grob from string (using the minifont) and embeds it at specified position (#x, #y). The grob is centered around #x and the to is put at #y. #w represents the maximum width of the grob created. If the text is wider, it is truncated. Bangtype.
3A4AB	MakeLabel	(\$ #w #x grob → grob') Inserts \$ into grob using CENTER\$3x5 with y=5.

4.11 Plotting

5127E	('IDPAR)	(→ id) Puts ID PPAR unevaluated on the stack. -- <REF>TEXT:Reserved PPAR
51166	CHECKPICT	(→) Checks size of GBUFF. If it is smaller than 131x64 sets GBUFF back to its default size (131x64).
51148	CKPICT	(xPICT →) Checks for user word xPICT on level 1. Errors (SETTYPEERR) if there is another object.

20CAD	PICTRCL	(xPICT → grob) Does CKPICT, then recalls GBUFF and does TOTEMPOB.
4AAEA	MAKEPVAR	(→ {}) Creates the default PPAR variable in the current directory and returns its value. -- <REF>TEXT:Reserved PPAR
4A9AF	CHECKPVAR	(→ {}) Recalls contents of PPAR in current path to stack. Creates PPAR in current directory if non-existent. Errors "Invalid PPAR" if existing PPAR is invalid. -- <REF>TEXT:Reserved PPAR
4B364	GETPARAM	(# → ob) Extracts the #th item from PPAR. No error checking! -- <REF>TEXT:Reserved PPAR
4B10C	GETXMIN	(→ %) Recalls XMIN from the PPAR list if existent. If not, the default PPAR is created in the current directory. -- <REF>TEXT:Reserved PPAR
4B166	PUTXMIN	(% →) Sets a new value for XMIN. PPAR is created if necessary. -- <REF>TEXT:Reserved PPAR
4B139	GETXMAX	(→ %) Recalls XMAX from the PPAR list if existent. If not, the default PPAR is created in the current directory. -- <REF>TEXT:Reserved PPAR
4B1AC	PUTXMAX	(% →) Sets a new value for XMAX. PPAR is created if necessary. -- <REF>TEXT:Reserved PPAR
4B120	GETYMIN	(→ %) Recalls YMIN from the PPAR list if existent. If not, the default PPAR is created in the current directory. -- <REF>TEXT:Reserved PPAR

4B189	PUTYMIN	(% →) Sets a new value for YMIN. PPAR is created if necessary. -- <REF>TEXT:Reserved PPAR
4B14D	GETYMAX	(→ %) Recalls YMAX from the PPAR list if existent. If not, the default PPAR is created in the current directory. -- <REF>TEXT:Reserved PPAR
4B1CF	PUTYMAX	(% →) Sets a new value for YMAX. PPAR is created if necessary. -- <REF>TEXT:Reserved PPAR
4B0DA	GETPMIN&MAX	(→ C% C%) -- Returns PMIN and PMAX. -- <REF>TEXT:Reserved PPAR
4AF77	PUTINDEP	(ID →) Internal xINDEP if the arg is an ID.
4AF8B	PUTINDEPLIST	({ } →) Internal xINDEP if the arg is a list.
510AD	INDEPVAR	(→ id) Recalls the independent variable. If a list, extract first element. :: GETINDEP DUPTYPELIST? ?CARCOMP ;
4AF63	GETINDEP	(→ id) (→ { }) Recalls the independent variable field in PPAR. -- <REF>TEXT:Reserved PPAR
4B062	GETPTYPE	(→ name) Recalls the plot type using GETPARAM. -- <REF>TEXT:Reserved PPAR
4B076	PUTPTYPE	(name →) Sets a new plot type. PPAR is created if necessary. -- <REF>TEXT:Reserved PPAR
4AFDB	GETRES	(→ %) Recalls the plot resolution using GETPARAM. -- <REF>TEXT:Reserved PPAR

4B012	PUTRES	(% →) Set new plot resolution. PPAR is created if necessary. -- <REF>TEXT:Reserved PPAR
4ADB0	GETSCALE	(→ % %') Recalls the plot scale parameters. -- <REF>TEXT:Reserved PPAR
4AE3C	PUTSCALE	(% %' →) Set new plot scale. PPAR is created if necessary. -- <REF>TEXT:Reserved PPAR
491D5	AUTOSCALE	(→) Internal AUTO.
4CE6F	DOGRAPHIC	(→) Sets the scroll mode of PICTURE and is essentially the same as { } PVIEW.
505C6	GETXPOS	
505E4	getxpos	
5068D	GETYPOS	
506AB	getypos	
15744	EQUATION	(→ ob T) (→ F) Recall the current equation, stored in the 'EQ' variable, and TRUE. If there is no 'EQ' variable on the path, just returns FALSE.
4A0AA	GetEqN	(#n → ob T) (#n → NULL\$ F) Get the #nth equation, if EQ is a list of equations.
1572B	DORCLE	(→ ob) Recalls the contents of the EQ variable, errors if it does not exist.
15717	DOSTOE	(ob →) Stores ob into the variable EQ.
20F8A	XEQPURGEPICT	(xPICT →) If object in level one is xPICT, erases the graphic display. Otherwise, errors.
00113	CRER	
4ECAD	CROSSMARKON	
4DA0D	CROSS_HAIRS	
4DA76	CROSS_OFF	
4CF05	GDISPCENTER	(→) Moves to center of graphics display
4B7D8	GetRes	

4B5AD	HSCALE	
4B553	VSCALE	
4B6D9	PLOTERR	
50DA5	PlotOneMore?	
4B765	PLOTPREP	
4F0AC	DOPX>C	({ hxs hxs' } → C%) Converts a list of two hex strings into a complex number. Used for plotting coordinates. Inverse operation is DOC>PX.
4F179	DOC>PX	(C% → { hxs hxs' }) Converts a complex coordinate point into list of two HXS numbers. Inverse operation is DOPX>C.

5 The HP49G CAS

5.1 Matrix Operations

5.1.1 Other Matrix Operations

6 UserRPL Commands

6.1 A-F

1AA1F	xABS	<p>(x → x') Absolute Value Function -- Returns the absolute value of its argument. x → x (x,y) → sqrt(x^2+y^2) x_unit → x _unit [array] → array 'sym' → 'ABS(sym)' -- Flags: -3 -- Related: NEG,SIGN</p>
1987E	xACK	<p>(→) Acknowledge Alarm cmd -- Acknowledges the oldest past due alarm. -- Flags: -43 -44 Repeat Alarms Not Rescheduled -43 Acknowledge Alarms Saved -44 -- Clears alert annunciator if 1. There are no other past-due alarms and 2. There are no other active alert sources - ie low batt. Has no effect on control alarms Control alarms that come due are automatically acknowledged AND saved in the sys alarm list. -- Related: ACKALL</p>

19863	xACKALL	<p>(\rightarrow) Acknowledge All Alarms cmd -- Acknowledges all past due alarms. -- Flags: -43 -44 Repeat Alarms Not Rescheduled -43 Acknowledge Alarms Saved -44 -- Clears alert annunciator if there are no other active alert sources, ie low batt. Has no effect on control alarms Control alarms that come due are automatically acknowledged AND saved in the sys alarm list. --</p>
1B72F	xACOS	<p>Related: ACK ($x \rightarrow x'$) Arc cos fn -- Returns angle with given cos. -- $z \rightarrow \text{arc cos } z$ 'sym' \rightarrow 'ACOS(sym)' --</p>
1B830	xACOSH	<p>Related: ASIN,ATAN,COS,ISOL,ACOSH ($x \rightarrow x'$) Arc hyp cos fn -- Returns val with given hyp cos. -- $z \rightarrow \text{arc hyp cos } z$ 'sym' \rightarrow 'ACOSH(sym)' --</p>
1BA3D	xALOG	<p>Related: ASINH,ATANH,COSH,ISOL ($x \rightarrow x'$) Common antilog fn -- ALOG $x = 10^x$ -- Flags: -3 numeric result -- $z \rightarrow 10^z$ 'sym' \rightarrow 'ALOG(sym)' -- Related: EXP,LN,LOG</p>

1E783	xAND	<p>(x1 x2 → x3) And fn -- Logical AND of 2 args. -- #n1 #n1 → #n3 "str1" "str2" → "str3" T/F1 T/F2 → 0/1 T/F 'sym' → 'T/F AND sym' 'sym' T/F → 'sym AND T/F' 'sym1' 'sym2' → 'sym1 AND sym2' -- Flags: -3 -5 Numeric res -3 Bin int wordsize -5 → -10 -- Related: NOT,OR,XOR</p>
1F5C5	xAPPLY	<p>({symb1 .. symbn} f → f(symb1...symbn)) Apply to args fn -- Creates expr for specified fn name & args -- Related: QUOTE, </p>
1E5D2	xARC	<p>(c r θ1 θ2 →) ({#x #y} #r θ1 θ2 →) Draw arc fn -- Draws arc in PICT anticlockwise from θ1 to θ2 centred on coord specified on lev4 with radius on lev3 -- Flags: -17 -18 angle mode (-17 & -18) -- Related: BOX,LINE,TLINE</p>

2125A	xARCHIVE	<pre>(:port:name →) (:IO:name →) Archive HOME cmd -- Creates backup of HOME in RAM (including user key assignments & alarm catalog) -- if :IO: is used backup transmitted through IO port via Kermit to filename 'name' -- Flags: -33 -39 I/O Device -33 I/O Messages -39 if :IO:name --</pre>
1B2DB	xARG	<pre>Related: RESTORE (c → θ) Argument fn -- Returns angle of a complex number -- (x,y) → θ 'sym' → 'ARG(sym)' -- Flags: -17 -18 Ang Mode -17,-18</pre>
1D092	xARRAY>	<pre>([] → x1...xn {n}) ([[]] → x11...xnm {n m}) Array to stack cmd -- Return elems of array to stack. OBJ→ includes this functionality. -- Related: →ARRAY,DTAG,EQ→,LIST→, OBJ→,STR→ UserRPL: xARRAY→</pre>
1D009	x>ARRAY	<pre>(x1...xn n → []) (x11...xnm {n m} → [[]]) Stack to Array Cmd -- Returns a vector of n real or complex elements or a matrix of n m real or complex solutions. -- Related: ARRAY→,LIST→,→LIST, OBJ→,STR→,→TAG,→UNIT UserRPL: x→ARRAY</pre>

```

1B6A4      xASIN      ( x → x' )
                Arc sin fn
                --
                Gives angle whose sin is given
                --
                z      → arc sin z
                'sym' → 'ASIN(sym)'
                --
                Flags: -1 -3 -17 -18
                Principal soln -1
                Numerical res  -3
                Angle mode     -17,-18
                --
                Related: ACOS,ATAN,ISOL,SIN
1B7EB      xASINH     ( x → x' )
                Arc hyp sin fn
                --
                Gives Val whose hyp sin is given
                --
                z      → arc hyp sin z
                'sym' → 'ASINH(sym)'
                --
                Flags: -1 -3
                Principal soln -1
                Numerical res  -3
                --
                Related: ACOSH,ATANH,ISOL,SINH

```

```

224F4      xASN      ( obj key → )
                ( 'SKEY' → )
                Assign cmd
                --
                Defines single key on user kbd by assigning the
                given obj to the key x_key
                --
                Flags: -61 -62
                User mode lock -61
                User mode      -62
                --
                The arg x_key is a real number rc.p where
                r=row,c=col,p=plane as follows:
                0,1 - unshifted
                2   - left shifted
                3   - right shifted
                4   - shifted
                5   - left shifted
                6   - right shifted
                Add 0.01 if the modifier is to be held pressed down.
                --
                After ASN, pressing the assigned in User or 1-User
                mode exeutes the assigned obj instead. Remains
                in effect until altered by ASN or STOKEYS or
                DELKEYS. If 'SKEY' is passed instead, the
                specified key is restored to std.
                --
                Related:      DELKEYS,RCLKEYS,STOKEYS
                <REF>TEXT:Keycodes
1957B      xASR      ( # → #' )
                Arithmetic shift right cmd
                --
                Shifts a bint 1 bit to the right except for the most
                significant bit which stays.
                --
                Flags: -5 -6 -7 -8 -9 -10 -11 -12
                bint wordsize  -5 -> -10
                bint base      -11, -12
                --
                Related: SL,SLB,SR,SRB

```

```

1B79C      xATAN      ( x → x' )
              Arc tan fn
              --
              Returns the angle having the tan
              --
              z      → arc tan z
              'sym' → 'ATAN(sym)'
              --
              Flags: -1 -3 -17 -18
              Principle soln -1
              Numeric results -3
              Angle mode      -17,-18
              --
              Related: ACOS,ASIN,ISOL,TAN
1B8A2      xATANH     ( x → x' )
              Arc hyp tan fn
              --
              Returns the value with given hyp tan.
              --
              z      → arc hyp tan z
              'sym' → 'ATANH(sym)'
              --
              Flags: -1 -3 -22
              Principle soln -1
              Numeric results -3
              Infinite result exception -22
              --
              Related: ACOSH,ASINH,ISOL,TANH
21448      xATTACH    ( n → )
              ( :nport:n → )
              Attach library cmd
              --
              Attaches lib with given num to current directory.
              --
              Related: DETACH,LIBS

```

```

1E1AB      xAUTO      ( → )
                  Calculates a y-axis display range
                  or an x- & y-axis display range.
                  --
                  Action depends on plot type:
FUNCTION      sets range to max &
              min of y vals sampled
              at 40 equi-spaced x
              vals (excluding )
CONIC        sets y-axis scale = to
              x-axis scale
POLAR        same as FUNCTION
;
PARAMETRIC  same as POLAR
;
TRUTH        no action
;
BAR          sets x-axis range from
              0 to # of elems in
              ΣDAT +1. sets y-range
              to min & max of the
              elts x-axis is always
              included.
HISTOGRAM    sets x-axis range to
              min & max of the elems
              in ΣDAT. sets y-range
              from 0 to # of rows in
              ΣDAT.
SCATTER      x-range is min & max
              of XCOL. y-range is
              min & max of YCOL
--
Related:      DRAW,SCALEH,SCALE,SCLΣ,
SCALEW,XRNG,YRNG
1E0BE      xAXES      ( c → )
                  ( {c tick $x $y } → )
                  Axes cmd
                  --
                  Specifies intersection coords of x- & y- axes, tick
                  mark annotatn and x- & y- axes labels. stored in
                  PPAR.
                  --
                  <REF>TEXT:Reserved|PPAR
                  --
                  Related: ATICK,DRAW,DRAX,LABEL

```



```

1E741      xBAR      ( → )
                Bar plot type cmd
                --
                Sets plot type to BAR. When plot type is BAR, the
                DRAW Cmd plots a bar chart using data from 1 col
                of the stat matrix (ΣDAT). The col to be plotted is
                specified by the XCOL cmd & is stored in 1st param
                of ΣPAR. Plot params are specified in PPAR of ff
                form:
                { (xmin,ymin) (xmax,ymax) indep
                  res axes ptype depend }
                For BAR they are used as follows:
                --
                (xmin,ymin) specifies lower left cnr of PICT
                (default: (-6.5,-3.1))
                --
                (xmax,ymax) specifies upper right cnr of PICT
                (default: (6.5,3.2))
                --
                indep name - specifies horiz axis label or list - {
                name x1 x2 } smaller of x1 & x2 is horiz location of
                1st bar (default: X)
                --
                res real - bar width in user units or bint - bar width
                in pixels (default: 0 - 1 in user units)
                --
                axes list containing one or more of the ff in order:
                (x1,y1) - user unit origin pos a list specifying tick
                mark annotatn & 2 strings specifying horiz & vert
                axes labels (default: (0,0))
                --
                ptype plot type - BAR in this case
                --
                depend label for vert axis. (default: Y)
                --
                <REF>TEXT:Reserved|PPAR
                --
                Related: CONIC,DIFFEQ,FUNCTION,GRIDMAP,
                HISTOGRAM,PARAMETRIC,PARSURFACE,PCONTOUR,
                SCATTER,SLOPEFIELD,TRUTH,YSLICE

```

20133	xBARPLOT	<p>(\rightarrow) Draw bar plot cmd -- Draws bar chart of specified col of stat matrix (ΣDAT) Col to be plotted is specified by XCOL & is stored as first param in ΣPAR. Default col is 1. data can be +ve or -ve giving bars above or below the axis. y-axis is autoscaled & plot type is BAR. When executed from a program, plot doesn't persist unless PICTURE,PVIEW (with empty list) or FREEZE is subsequently executed -- Related: FREEZE,HISTPLOT,PICTURE,PVIEW,SCATRLOT,XCOL</p>
2200C	xBAUD	<p>(n \rightarrow) Baud rate cmd -- Specify bit transfer rate. -- Related: CKSM,PARITY,TRANSIO</p>
1A5C4	xBEEP	<p>(freq dur \rightarrow) Beep cmd -- Sounds a tone of n Hz for x secs. -- Flags: -56 Error Beep -56 Max Freq = 4400 Hz Max Duration = 1048.575 secs. -- Related: HALT,INPUT,PROMPT,WAIT</p>
2025E	xBESTFIT	<p>(\rightarrow) Best fit model cmd -- Executes LR with each of the 4 curve fitting models and selects the model giving the largest correlation coefficient. -- Selected model stored in 5th param of ΣPAR & regression coeffs intercept & slope are stored in 3rd & 4th params. -- Related: EXPFIT,LINFIT,LOGFIT,LR,PWRFIT</p>

1C559	xBIN	<p>(→) Binary mode cmd -- Selects binary base for bint ops. (Default base is 10) -- Flags: -5 -6 -7 -8 -9 -10 -11 -12 Bint wordsize -5 → -10 Bint base -11, -12 Bints require prefix #. Bints entered & returned in binary show the b suffix. If current base not binary, enter binary nums by using b suffix. The current base doesn't affect the internal representation of bints as unsigned bints. --</p>
2010E	xBINS	<p>Related: DEC,HEX,OCT,STWS,RCWS (min width n → [[]] []) Sort Into Frequency Bins Cmd -- Sorts the elements of the indep. col (XCOL) of the stat matrix (ΣDAT) into (nbins + 2) bins, where the left edge for bin 1 starts at value xmin and each bin has width xwidth. -- xmin xwidth nbins → [[nbin1...nbinn]] [nbinL nbinR] --</p>
1E416	xBLANK	<p>Related: BARPLOT,XCOL (#width #height → grob) Blank Graphics Obj Cmd -- Creates a blank graphics obj of the specified width and height. --</p>
1E3EC	xBOX	<p>Related: →GROB,LCD→ ({#n1 #m1} {#n2 #m2} →) (c1 c2 →) Box Cmd -- Draws in PICT a box whose opposite corners are defined by the specified pixel or user-unit coords. -- Related: ARC,LINE,TLINE</p>

22087	xBUFLN	(\rightarrow nchars 0/1) Buffer Length Cmd -- Returns the number of characters in the HP 48's serial input buffer and a single digit indicating whether an error occurred during data reception. -- Related: CLOSEIO,OPENIO,SBRK,SRECV,STIME,XMIT
1A1D9	xBYTES	(obj \rightarrow chksum size) Bytes Size Cmd -- Returns the number of bytes & the checksum for the given obj. -- Related: MEM
196BB	xB>R	(# \rightarrow R) Binary to Real Cmd -- Converts a binary integer to its floating-point equivalent. -- Related: R \rightarrow B UserRPL: xB \rightarrow R
2378D	xCASE	(\rightarrow) CASE Conditional Structure Cmd -- Starts CASE ... END conditional structure. -- CASE \rightarrow THEN T/F \rightarrow END \rightarrow END \rightarrow --
1BC0F	xCEIL	Related: END,IF,IFERR,THEN (x \rightarrow n) Ceiling Func -- Returns the smallest integer greater than or equal to the argument. -- x \rightarrow n x_u \rightarrow n_u 'sym' \rightarrow 'CEIL(sym)' -- Flags: -3 -- Related: FLOOR,IP,RND,TRNC

1E0E8	xCENTR	<p>((x,y) →) (x →) Centre Cmd -- Adjusts the first two parameters in the reserved variable PPAR, (xmin, ymin) and (xmax,ymax), so that the point represented by the argument (x,y) is the plot centre. -- <REF>TEXT:Reserved PPAR --</p>
1C2D5	xCF	<p>Related: SCALE (n →) Clear Flag Cmd -- Clears the specified user or system flag. --</p>
1CB66	xCHR	<p>Related: FC?,FC?C,FS?,FS?C,SF (n → \$) Character Cmd -- Returns a string representing the HP 48 character corresponding to the character code n. --</p>
1C149	x%CH	<p>Related: NUM,POS,REPL,SIZE,SUB (x1 x2 → x3) Percent Change Func -- Returns the percent change from x (level 2) to y (level 1) as a percentage of x. -- $x \quad y \quad \rightarrow 100(y-x)/x$ $x \quad 'sym' \rightarrow '%CH(x,sym)'$ $'sym' \quad x \quad \rightarrow '%CH(sym,x)'$ $'sym1' \quad 'sym2' \rightarrow '%CH(sym1,sym2)'$ $x_u \quad y_u \quad \rightarrow 100(y_u-x_u)/x_u$ $x_u \quad 'sym' \rightarrow '%CH(x_u,sym)'$ $'sym' \quad x_u \quad \rightarrow '%CH(sym,x_u)'$ -- Flags: -3 -- Related: %,%T </p>

21FEC	xCKSM	(<i>n_type</i> →) Checksum Cmd -- Specifies the error-detection scheme. --
1FCEB	xCLEAR	Related: BAUD,PARITY,TRANSIO ; (<i>ob1</i> .. <i>obn</i> →) Clear Cmd -- Removes all objects from the stack. --
1FD2B	xCLSIGMA	Related: CLVAR,PURGE (→) Clear Sigma Cmd -- Purges the current statistics matrix (reserved variable Σ DAT). -- <REF>TEXT:Reserved Σ DAT --
198DE	xCLKADJ	Related: RCL Σ ,STO Σ , Σ +, Σ - UserRPL: xCL Σ (<i>ticks</i> →) Adjust System Clock Cmd -- Adjusts the system time by x clock ticks, where 8192 clock ticks equal 1 second. --
1A858	xCLLCD	Related: →TIME (→) Clear LCD Cmd -- Clears (blanks) the stack display --
21ED5	xCLOSEIO	Related: DISP,FREEZE (→) Close I/O Port Cmd -- Closes the serial port and the IR port, and clears the input buffer and any error messages for KERMIT. -- Related: BUFLLEN,OPENIO

210FC	xCLUSR	(\rightarrow) Clear Variables Cmd -- Purges all variables and empty subdirectories in the current directory. -- Related: CLUSR,PGDIR,PURGE UserRPL: xCLVAR
1BFBE	xCNRM	([] \rightarrow col_norm) Column Norm Cmd -- Returns the column norm (onenorm) of the array argument. -- Related: CROSS,DET,DOT,RNRM
20A15	xCOLCT	(symb \rightarrow symb') Collect Like Terms Cmd -- Simplifies an algebraic expression or equation by "collecting" like terms. Does not modify numbers. -- Related: EXPAN,ISOL,QUAD,SHOW
2009A	xSIGMACOL	(x_col y_col \rightarrow) Sigma Columns Cmd -- Specifies the independent variable and dependent-variable columns of the current stat matrix (the reserved variable Σ DAT). -- <REF>TEXT:Reserved Σ DAT -- Related: BARPLOT,BESTFIT,CORR,COV,EXPFIT,HISTPLOT,LINFIT,LOGFIT,LR,PREDX,PREDY,PWRFIT,SCATRLOT,XCOL,YCOL UserRPL: xCOL Σ
1C1F6	xCOMB	(n k \rightarrow Cn,k) Combinations Func -- Returns the number of possible combinations of n items taken m at a time. -- n m \rightarrow Cn:m 'symn' m \rightarrow 'COMB(symn,m)' n 'symm' \rightarrow 'COMB(n,symm)' 'symn' 'symm' \rightarrow 'COMB(symn,symm)' -- Related: PERM,!

1D186	xCON	<p>({ n } x → []) ({ n k } x → [[]]) ([] x → [] ') Constant Array Cmd -- Returns a constant array, defined as an array whose elements all have the same value. -- {ncols} zcnst → [[vec cnst]] {nrows mrows} zcnst → [[mat cnst]] [R-arr] xcnst → [R-arr cnst] [C-arr] xcnst → [C-arr cnst] 'name' zcnst → --</p>
1E681	xCONIC	<p>Related: IDN (→) Conic Plot Type Cmd -- Sets the plot type to CONIC. -- Related: BAR, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE</p>
1AA6E	xCONJ	<p>(x → x') Conjugate Analytic Func -- Conjugates a complex number or a complex array. -- x → x (x, y) → (x, -y) [R-arr] → [R-arr] [C-arr]1 → [C-arr]2 'sym' → 'CONJ(sym)' -- Flags: -3 --</p>
1A8BB	xCONT	<p>Related: ABS, IM, RE, SCONJ, SIGN (→) Continue Program Execution Cmd -- Resumes execution of a halted program. -- Related: HALT, KILL, PROMPT</p>

196DB	xCONVERT	<p>(x1_u1 x2_u2 → x3_u2) Convert Units Cmd -- Converts a source unit object to the dimensions of a target object -- Related: UBASE,UFACT,→UNIT,UVAL</p>
1FDC1	xCORR	<p>(→ x_correlation) Correlation Cmd -- Returns the correlation coefficient of the independent and dependent data columns in the current statistics matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved ΣDAT -- Related: COLΣ,COV,PREDX,PREDY,XCOL,YCOL</p>
1B505	xCOS	<p>(x → x') Cos Func -- Returns the cos of the argument. -- z → cos z 'sym' → 'COS(sym)' x_uangular → cos(x_uangular) -- Flags: -3 -17 -18 --</p>
1B606	xCOSH	<p>Related: ACOS,SIN,TAN (x → x') Hyp Cos Func -- Returns the hyp cos of the argument. -- z → cosh z 'sym' → 'COSH(sym)' -- Flags: -3 -- Related: ACOSH,SINH,TANH</p>

1FDCC	xCOV	(\rightarrow x_covariance) Covariance Cmd -- Returns the sample covariance of the independent and dependent data columns in the current stat matrix (reserved variable Σ DAT). -- <REF>TEXT:Reserved Σ DAT -- Related: COL Σ ,CORR,PCOV,PREDX,PREDY, XCOL,YCOL
1EEA4	xCR	(\rightarrow) Carriage Right Cmd -- Prints the contents, if any, of the printer buffer. -- Flags: -37 -34 -33 -- Related: DELAY,OLDPRT,PRLCD,PRST,PRSTC, PRVAR,PR1
1A105	xCRDIR	(name \rightarrow) Create Directory Cmd -- Creates an empty subdirectory with the specified name within the current directory. -- Related: HOME,PATH,PGDIR,UPDIR
1C01E	xCROSS	([1] [2] \rightarrow [3]) Cross Product Cmd -- CROSS returns the cross product [3] = [1] x [2] of vectors [1] and [2]. -- Related: CNRM,DET,DOT,RNRM
1E29A	xC>PX	((x,y) \rightarrow {#n #m}) Complex to Pixel Cmd -- Converts the specific user-unit coordinates to pixel coordinates. -- (x,y) \rightarrow { #n #m } -- Related: PX \rightarrow C UserRPL: xC \rightarrow PX

1C98E	<code>xC>R</code>	<p>((x,y) → x y) ([C] → [R] [I]) Complex to Real Cmd -- Separates the real and imaginary parts of a complex number or complex array. -- Related: R→C,RE,IM UserRPL: xC→R</p>
19812	<code>xDATE</code>	<p>(→ date) Returns the system date. -- Related: DATE+,DDAYS,TIME,TSTR</p>
1989E	<code>xSETDATE</code>	<p>(date →) Set Date Cmd -- Sets the system date to date. -- Related: →TIME UserRPL: x→DATE</p>
199D2	<code>xDATE+</code>	<p>(date ndays → date') Date Addition Cmd -- Returns a past or future date, given a date in level 2 and a number of days in level 1. -- Flags: -42 -- Related: DATE,DDAYS</p>
199B2	<code>xDDAYS</code>	<p>(date1 date2 → days) Delta Days Cmd -- Returns the number of days between two dates. -- Related: DATE,DATE+</p>
1C574	<code>xDEC</code>	<p>(→) Decimal Mode Cmd -- Selects decimal base for binary integer operations. (The default base is decimal.) -- Related: BIN,HEX,OCT,RCWS,STWS</p>

209AA	xDECR	<p>(name \rightarrow x_new) Decrement Cmd -- Takes a variable on level 1, subtracts 1, stores the new value back into the original variable, and returns the new value to level 1. --</p>
20D65	xDEFINE	<p>Related: INCR,STO+,STO- ('name=expr' \rightarrow) ('name(name1...)=expr(name1...)' \rightarrow) Define Variable or Func Cmd -- Stores the expression on the right side of the = in the variable specified on the left side, or creates a user-defined function --</p>
1C399	xDEG	<p>Related: STO (\rightarrow) Degrees Cmd -- Sets Degrees angle mode. --</p>
19972	xDELALARM	<p>Related: GRAD,RAD (n \rightarrow) Delete Alarm Cmd -- Deletes the alarm specified in level 1. -- Related: FINDALARM,RCLALARM,STOALARM</p>
1EF43	xDELAY	<p>(x_delay \rightarrow) Delay Cmd -- Specifies how many seconds the HP 48 waits between sending lines of information to the printer. --</p>
22548	xDELKEYS	<p>Related: CR,OLDPRT,PRLCD,PRST,PRSTC,PRVAR,PR1 (rc.p \rightarrow) ({ rc.p ... n } \rightarrow) Delete Key Assignments Cmd -- Clears user-defined key assignments. -- Related: ASB,RCLKEYS,STOKEYS</p>

1E22B	xDEPND	(name →) ({name} →) ({name y1 y2} →) ({y1 y2} →) (y1 y2 →) Dependent Variable Cmd -- Species the dependent variable (and its plotting range for TRUTH plots). --
1FC44	xDEPTH	Related: INDEP (→ n) Depth Cmd -- Returns a real number representing the number of objects present on the stack (before DEPTH was executed).
1BFDE	xDET	([[]] → x) Determinant Func -- Returns the determinant of a square matrix. --
2147C	xDETACH	Related: CNRM,CROSS,DOT,RNRM (n →) (:port:n →) Detach Library Cmd -- Detaches the library with the specified number from the current directory. Each library has a unique number. If a port number is specified, it is ignored. --
23813	xDIR	Related: ATTACH,LIBS,PURGE
1A584	xDISP	(obj n_line →) Display Cmd -- Displays obj in the nth display line. -- Related: FREEZE,HALT,INPUT,PROMPT

230C3	xD0	<p>(→) DO Indefinite Loop Structure Cmd -- Starts DO ... UNTIL ... END indefinite loop structure. -- DO → UNTIL → END T/F → -- Related: END,UNTIL,WHILE</p>
1A339	xDOERR	<p>(n →) (#n →) (\$ →) (0 →) Do Error Cmd -- Executes a "user-specified" error, causing a program to behave exactly as if a normal error had occurred during program execution. -- Related: ERRM,ERRN,ERRO</p>
1BFFE	xDOT	<p>([1] [2] → x) Dot Product Cmd -- Returns the dot product AoB of two arrays A and B, calculated as the sum of the products of the corresponding elements of the two arrays. -- Related: CNRM,CROSS,DET,RNRM</p>
1E190	xDRAW	<p>(→) Draw Plot Cmd -- Plots the mathematical data in the reserved variable EQ or the statistical data in the reserved variable ΣDAT, using the specified x- and y-axis display ranges. -- <REF>TEXT:Reserved EQ -- Related: AUTO,AXES,DRAX,ERASE,FREEZE, PICTURE,LABEL,PVIEW</p>

1E1C6	xDRAX	(→) Draw Axes Cmd -- Draws axes in PICT. -- Related: AXES,DRAW,LABEL
1FBDB	xDROP	(ob →) Drop Object Cmd -- Removes the level 1 object from the stack. -- Related: CLEAR,DROPN,DROP2
1FBF3	xDROP2	(ob1 ob2 →) Drop 2 Objects Cmd -- Removes the first two objects from the stack. -- Related: CLEAR,DROP,DROPN
1FC64	xDROPN	(ob1 . . . obn n →) Drop n Objects Cmd -- Removes the first n + 1 objects from the stack (the first n objects excluding the integer n itself). -- Related: CLEAR,DROP,DROP2
22633	xDTAG	(tag:obj → obj) Delete Tag Cmd -- DTAG removes all tags (labels) from an object. -- Related: LIST→,→TAG
1FB87	xDUP	(ob → ob ob) Duplicate Object Cmd -- DUP returns a copy to level 1 of the object in level 1. -- Related: DUPN,DUP2,PICK
1FBA2	xDUP2	(1 2 → 1 2 1 2) Duplicate 2 Objects Cmd -- DUP2 returns copies of the objects in levels 1 and 2 of the stack. -- Related: DUP,DUPN,PICK

1FC7F	xDUPN	<p>(1 . . . n n \rightarrow 1 . . . n 1 . . . n) Duplicate n Objects Cmd -- Takes an integer n from level 1 of the stack, and returns copies of the objects in stack levels 2n through n + 1. --</p>
1BEC8	xD>R	<p>Related: DUP,DUP2,PICK (x \rightarrow ($\pi/180$) x) Degrees to Radians Func -- Converts a real number representing an angle in degrees to its equivalent in radians. -- x \rightarrow ($\pi/180$) x 'sym' \rightarrow 'D\rightarrowR(sym)' --</p>
1AB23	xCONSTANTe	<p>Related: R\rightarrowD UserRPL: xD\rightarrowR (\rightarrow e) e Func -- Returns the symbolic constant e or its numerical representation, 2.71828182846. --</p>
22FB5	xELSE	<p>Related: EXP,EXPM,i,LN,LNP1,MAXR,MINR,π UserRPL: xe (\rightarrow) ELSE Cmd -- Starts false clause in conditional or error-trapping structure. See the IF and IFERR keyword entries for syntax information. --</p>
236B9	xENDDO	<p>Related: IF,CASE,DO,ELSE,IFERR,REPEAT,THEN,UNTIL,WHILE (1/0 \rightarrow) END Cmd -- Ends conditional, error-trapping, and indefinite loop structures. ; See the IF, CASE, IFERR, DO, and WHILE keyword entries for syntax information. -- Related: IF,CASE,DO,ELSE,IFERR,REPEAT,THEN,UNTIL,WHILE UserRPL: xEND</p>

22FD5	xIFEND	<p>END Cmd</p> <p>--</p> <p>Ends conditional, error-trapping, and indefinite loop structures.</p> <p>--</p> <p>See the IF, CASE, IFERR, DO, and WHILE keyword entries for syntax information.</p> <p>--</p> <p>Related: IF,CASE,DO,ELSE,IFERR,REPEAT,THEN,UNTIL,WHILE UserRPL: xEND</p>
23694	xWHILEEND	<p>END Cmd</p> <p>--</p> <p>Ends conditional, error-trapping, and indefinite loop structures.</p> <p>--</p> <p>See the IF, CASE, IFERR, DO, and WHILE keyword entries for syntax information.</p> <p>--</p> <p>Related: IF,CASE,DO,ELSE,IFERR,REPEAT,THEN,UNTIL,WHILE UserRPL: xEND</p>
1C452	xENG	<p>(n →)</p> <p>Engineering Mode Cmd</p> <p>--</p> <p>Sets the number display format to Engineering mode, which displays one to three digits to the left of the fraction mark (decimal point) and an exponent that is a multiple of three. The total number of significant digits displayed is $n + 1$.</p> <p>--</p> <p>Related: FIX,SCI,STD</p>
1CEE3	xEQ>	<p>('l=r' → l r)</p> <p>Equation to Stack Cmd</p> <p>--</p> <p>Separates an equation into its left and right sides.</p> <p>--</p> <p>'sym1=sym2' → 'sym1' 'sym2'</p> <p>z → z 0</p> <p>'name' → 'name' 0</p> <p>x_u → x_u 0</p> <p>'sym' → 'sym' 0</p> <p>--</p> <p>Related: ARRAY→,DTAG,LIST→,OBJ→,STR→ UserRPL: xEQ→</p>

1E25F	xERASE	(→) Erase PICT Cmd -- Erases PICT, leaving a blank PICT of the same dimensions. --
1A36D	xERRO	Related: DRAW (→) Clear Last Error Number Cmd -- Clears the last error number so that a subsequent execution of ERRN returns # 0h, and clears the last error message. --
1A3A3	xERRM	Related: DOERR,ERRM,ERRN (→ \$msg) Error Message Cmd -- Returns a string containing the error message of the most recent calculator error. --
1A388	xERRN	Related: DOERR,ERRN,ERRO (→ \$nerr) Error Number Cmd -- Returns the error number of the most recent calculator error. -- Related: DOERR,ERRM,ERRO

1A3BE	xEVAL	<p>(ob \rightarrow ?) Evaluate Object Cmd -- Evaluates the object. -- obj \rightarrow (see below) Obj. Type Effects of Evaluation Local Name Recalls the contents of the variable. Global Name Calls the contents of the variable: ; A name is evaluated. A program is evaluated. A directory becomes the current directory. Other objects are put on the stack. If no variable exists for a given name, evaluating the name returns the name to the stack. Program. Enters each object in the program: Names are evaluated (unless quoted). ed). Cmds are evaluated. Other objects are put on the stack. List Enters each object in the list: Names are evaluated. Cmds are evaluated. Programs are evaluated. Other objects are put on the stack. Tagged If the tag specifies a port, recalls and evaluates the specified object. Otherwise, puts the untagged object on the stack. Algebraic Enters each object in the algebraic expression: Names are evaluated. Cmds are evaluated. Other objects are put on the stack. Cmd, Func, XLIB Name Evaluates the specified object. Other Objects Puts the object on the stack. --</p>
1B905	xEXP	<p>Related: \rightarrowNUM,SYSEVAL (x \rightarrow x') Exponential Analytic Func -- Returns the exponential, or natural antilogarithm, of the argument; that is, e raised to the given power. -- z \rightarrow ez 'sym' \rightarrow 'EXP(sym)' --</p>
20A49	xEXPAN	<p>Related: ALOG,EXPM,LN,LOG (symb1 \rightarrow symb2) Expand Products Cmd -- Rewrites an algebraic expression or equation by expanding products and powers. -- Related: COLCT,EXPAND,ISOL,QUAD,SHOW</p>

201FB	xEXPFIT	<p>(\rightarrow) Exponential Curve Fit Cmd -- Stores EXPFIT as the fifth parameter in the reserved variable ΣPAR, indicating that subsequent executions of LR are to use the exponential curve fitting model. -- <REF>TEXT:Reserved ΣPAR -- Related: BESTFIT,LR,LINFIT,LOGFIT,PWRFIT</p>
1BAC2	xEXPM	<p>($x \rightarrow x'$) Exponential Minus 1 Analytic Func -- Returns $e^x - 1$. -- $x \rightarrow e^x - 1$ 'sym' \rightarrow 'EXPM(sym)' -- Related: EXP,LNP1</p>
1C360	xFC?	<p>($n \rightarrow 0/1$) Flag Clear? Cmd -- Tests whether the system or user flag specified by nflag number is clear, and returns a corresponding test result: 1 (true) if the flag is clear or 0 (false) if the flag is set. -- Related: CF,FC?C,FS?,FS?C,SF</p>
1C520	xFC?C	<p>($n \rightarrow 0/1$) Flag Clear? Clear Cmd -- Tests whether the system or user flag specified by nflag number is clear, and returns a corresponding test result: 1 (true) if the flag is clear or 0 (false) if the flag is set. After testing, clears the flag. -- Related: CF,FC?,FS?,FS?C,SF</p>

19948	xFINDALARM	<p>(date → n) ({date time} → n) (0 → n) Find Alarm Cmd -- Returns the alarm index nindex of the first alarm due after the specified time. -- Related: DELALARM,RCLALARM,STOALARM</p>
21FB6	xFINISH	<p>(→) Finish Server Mode Cmd -- Terminates Kermit Server mode in a device connected to an HP 48. -- Related: BAUD,CKSM,KGET,PARITY,PKT,RECN,RECV,SEND,SERVER</p>
1C3EA	xFIX	<p>(n →) Fix Mode Cmd -- Sets the number display format to Fix mode, which rounds the display to n display places. -- Related: SCI,STD,ENG</p>
1BBD9	xFLOOR	<p>(x → n) Floor Func -- Returns the greatest integer that is less than or equal to the argument. -- x → n x_u → n_u 'sym' → 'FLOOR(sym)' -- Related: CEIL,IP,RND,TRNC</p>

231A0	xSTARTVAR	<pre>(start finish →) FOR Definite Loop Structure Cmd -- Starts FOR ... NEXT and FOR ... STEP definite loop structures. -- FOR xstart xfinish → NEXT → FOR xstart xfinish → STEP xincrement → STEP 'symincrement' → -- Related: NEXT,START,STEP UserRPL: xFOR</pre>
1BBA3	xFP	<pre>(x → x') Fractional part Func -- Returns the fractional part of an argument. -- x → y x_u → y_u 'sym' → 'FP(sym)'</pre>
213D1	xFREE	<pre>Related: IP Not useful on the 49G. Free RAM Card Cmd -- Frees (makes independent) the previously merged RAM in port 1. FREE is provided for compatibility with the HP 48SX, which could merge RAM in port 2 as well. See FREE1. -- { } nport → { namebackup ... nlib } nport → namebackup nport → nlib nport →</pre>
1A5A4	xFREEZE	<pre>(n →) Freeze Display Cmd -- Freezes the part of the display specified by ndisplay area, so that it is not updated until a key is pressed. -- Related: CLLCD,DISP,HALT</pre>

1C4A1	xFS?C	(n → 0/1) Flag Set? Clear Cmd -- Tests whether the system or user flag specified by nflag number is clear, and returns a corresponding test result: 1 (true) if the flag is set or 0 (false) if the flag is clear. After testing, clears the flag --
1C313	xFS?	Related: CF,FC?,FC?C,FS?C,SF (n → 0/1) Flag Set Cmd -- Tests whether the system or user flag specified by nflag number is set, and returns a corresponding test result: 1 (true) if the flag is set or 0 (false) if the flag is clear. --
1E661	xFUNCTION	Related: CF,FC?,FC?C,FS?,SF (→) Function Plot Type Cmd -- Sets the plot type to FUNCTION. -- Related: BAR,CONIC,DIFFEQ,FASTEQ,FAST3D,GRIDMAP,HISTOGRAM,PARAMETRIC,PARSURFACE,PCONTOUR,POLAR,SCATTER,SLOPEFIELD,TRUTH,WIREFRAME,YSLICE

6.2 G-M

1D7C6 xGET

(ob n → elm)

ob = [] or [[]] or {} or name

pos = n or {n} or {n m}

Get Element Command

--

Returns from the level 2 array or list (or named array or list) the real or complex number zget or object objget whose position is specified in level 1.

--

[[mat]] nposition → zget

[[mat]] { nrow mcol } → zget

'namematrix' nposition → zget

'namematrix' { nrow mcol } → zget

[vector] nposition → zget

[vector] { nposition } → zget

'namevector' nposition → zget

'namevector' { nposition } → zget

{ list } nposition → objget

{ list } {nposition} → objget

'namelist' nposition → objget

'namelist' {nposition} → objget

--

Related: GETI,PUT,PUTI

1D8C7 xGETI

(ob pos → ob' pos' elm)
 ob = [] or [[]] or {} or name
 pos = n or {n} or {n m}
 Get and Increment Index Command

--

Returns from the level 2 array or list (or named array or list) the real or complex number zget or object objget whose position is specified in level 1, along with the level 2 argument and the next position in that argument.

--

```
[[ mat ]] npos1
→ [[ mat ]] npos2 zget
[[ mat ]] { nr mc }1
→ [[ mat ]] { nr mc }2 zget
'namemat' npos1
→ 'namemat' npos2 zget
'namemat' { nr mc }1
→ 'namemat' { nr mc }2 zget
[ vec ] npos1
→ [ vec ] npos2 zget
[ vec ] { npos1 }
→ [ vec ] { npos2 } zget
'namevec' npos1
→ 'namevec' npos2 zget
'namevec' { npos1 }
→ 'namevec' { npos2 } zget
{ list } npos1
→ { list } npos2 objget
{ list } { npos1 }
→ { list } { npos2 } objget
'namelist' npos1
→ 'namelist' npos2 objget
'namelist' { npos1 }
→ 'namelist' { npos2 } objget
```

--

Related: GET,PUT,PUTI

1E456	xGOR	<pre>(g_targ {#n #m} grob → g_targ') (g_targ (x,y) grob → g_targ') (PICT ... →) Graphics OR Cmd -- Superimposes grob1 onto grobtgt or PICT, with the upper left corner of grob1 positioned at the specified coordinate in grobtgt or PICT. -- gobtgt {#n #m} grob1 → grob' gobtgt (x,y) grob1 → grob' PICT {#n #m} grob1 → PICT (x,y) grob1 → -- Related: GXOR,REPL,SUB</pre>
1C3CF	xGRAD	<pre>(→) Grads Mode Cmd -- Sets Grads angle mode. -- Related: GRAD,RAD</pre>
23813	xGROB	
1E5AD	x>GROB	<pre>(ob n_chrsiz → grob) Stack to Graphics Object Cmd -- Creates a graphics object representing the level 2 object, where the argument nchar size specifies the character size of the representation. -- Related: →LCD,LCD→ UserRPL: x→GROB</pre>
1E4E4	xGXOR	<pre>(g_targ {#n #m} g_src → g_targ') (g_targ (x,y) g_src → g_targ') (PICT ... →) Graphics Exclusive OR Cmd -- Superimposes grob1 onto grobtgt or PICT, with the upper left corner of grob1 positioned at the specified coordinate in grobtgt or PICT. -- gobtgt {#n #m} grob1 → grobresult gobtgt (x,y) grob1 → grobresult PICT {#n #m} grob1 → PICT (x,y) grob1 → -- Related: GOR,REPL,SUB</pre>

23472	xHALT	(→) Halt Program Cmd -- Halts program execution. -- Related: CONT,KILL
1C58F	xHEX	(→) Hexadecimal Mode Cmd -- Selects hexadecimal base for binary integer operations. (The default base is decimal.) -- Related: BIN,OCT,DEC,RCWS,STWS
1E721	xHISTOGRAM	(→) Histogram Plot Type Cmd -- Sets the plot type to HISTOGRAM. -- Related: BAR,CONIC,DIFFEQ,FUNCTION,GRIDMAP,PARAMETRIC,PARSURFACE,PCONTOUR,POLAR,SCATTER,SLOPEFIELD,TRUTH,WIREFRAME,YSLICE
20167	xHISTPLOT	(→) Draw Histogram Plot Cmd -- Plots a frequency histogram of the specified column in the current stat matrix (reserved matrix Σ DAT). -- <REF>TEXT:Reserved Σ DAT -- Related: BARPLOT,BINS,FREESE,PICTURE,PVIEW,RES,SCATRPLOT,XCOL
1BF7E	xHMS-	(hms1 hms2 → hms3) Hours-Minutes-Seconds Minus Cmd -- Returns the difference of two real number, where the arguments and the result are interpreted in hours-minutes-seconds format. -- Related: HMS→,→HMS,HMS+

1BF5E	xHMS+	(<i>hms1 hms2</i> → <i>hms3</i>) Hours-Minutes-Seconds Plus Cmd -- Returns the sum of two real number, where the arguments and the result are interpreted in hours-minutes-seconds format. -- Related: HMS→,→HMS,HMS-
1BF1E	x>HMS	(<i>x</i> → <i>x'</i>) Decimal to Hours-Minutes-Seconds Cmd -- Converts a real number representing hours or degrees with a decimal fraction to hours-minutes-seconds format. -- Related: HMS→,HMS+,HMS- UserRPL: x→HMS
1BF3E	xHMS>	(<i>x</i> → <i>x'</i>) Hours-Min-Sec to Decimal Cmd -- Converts a real number in hours -minutes-seconds format to its decimal form (hours or degrees with a decimal fraction). -- Related: HMS→,HMS+,HMS- UserRPL: xHMS→
1A140	xHOME	(→) HOME Directory Cmd -- Makes the HOME directory the current directory. -- Related: CRDIR,PATH,PGDIR,UPDIR
1AB45	xi	(→ <i>i</i>)
1D2DC	xIDN	(<i>n</i> → [[]]) ([[]] → [[]] ') (<i>name</i> → [[]]) Identity Matrix Cmd -- Returns an identity matrix; that is, a square matrix with its diagonal elements equal to 1 and its off-diagonal elements equal to 0. -- Related: CON

22EC3	xIF	<p>(→) IF Conditional Structure Cmd -- Starts IF ... THEN ... END and IF ... THEN ... ELSE ... END conditional structures. -- IF → THEN T/F → END → → IF → THEN T/F → ELSE → END → --</p>
233DF	xIFERR	<p>Related: CASE,ELSE,END,IFERR,THEN (→) If Error Conditional Struct Cmd -- Starts IFERR ... THEN ... END and IFERR ... THEN ... ELSE ... END error trapping structures. --</p>
1A4CD	xIFT	<p>Related: CASE,ELSE,END,IF,THEN (0/1 obj → ?) IF-THEN Cmd -- Executes obj if T/F is nonzero. Discards obj if T/F is zero. --</p>
1A3FE	xIFTE	<p>Related: IFTE (0/1 objT objF → ?) IF-THEN-ELSE Cmd -- Executes objT if T/F is nonzero. Discards objF if T/F is zero. --</p>
1AB45	xi	<p>Related: IFT (→ i)</p>

1C819	xIM	$((x,y) \rightarrow y)$ $([] \rightarrow []')$ Imaginary Part Func -- Returns the imaginary part of its (complex) argument. -- $x \rightarrow 0$ $(x,y) \rightarrow y$ $[R\text{-arr}] \rightarrow [R\text{-arr}]$ $[C\text{-arr}] \rightarrow [R\text{-arr}]$ $'sym' \rightarrow 'IM(sym)'$ -- Related: $C \rightarrow R, RE, R \rightarrow C$
208F4	xINCR	$(name \rightarrow x)$ Increment Cmd -- Takes a variable on level 1, adds 1, stores the new value back into the original variable, and returns the new value to level 1. -- Related: DECR
1E04A	xINDEP	$(name \rightarrow)$ $(\{name\} \rightarrow)$ Independent Variable Cmd -- Specifies the independent variable and its plotting range. -- Related: DEPND
224CA	xINPUT	$(\$prompt \$ \rightarrow \$')$ $(\$prompt \{specs\} \rightarrow \$')$ Input Cmd -- Prompts for data input to the command line and prevents the user access to stack operations. -- Related: PROMPT,STR \rightarrow
1B278	xINV	$(x \rightarrow 1/x)$ $([[]] \rightarrow [[]]')$ Inverse (1/x) Analytic Func -- Returns the reciprocal or the matrix inverse. -- Related: SINV,/

1BB6D	xIP	($x \rightarrow n$) Integer Part Func -- Returns the integer part of the argument. -- $x \rightarrow n$ $x_u \rightarrow n_u$ 'sym' \rightarrow 'IP(sym)' --
20A93	xISOL	Related: FP ($\text{symb var} \rightarrow \text{symb}'$) Isolate Variable Cmd -- Returns an algebraic symb' that rearranges symb to "isolate" the first occurrence of variable var . -- Related: COLCT,EXPAN,QUAD,SHOW,SOLVE
1FAEB	xFORMUNIT	UserRPL: $x_$
23654	x'	
23679	xENDTIC	UserRPL: x'
2361E	$x\ll$	UserRPL: $x\ll$
23639	$x\gg$	UserRPL: $x\gg$
235FE	$x\gg$ ABND	UserRPL: $x\gg$
2206C	xKERRM	($\rightarrow \text{msg}$) Kermit Error Message Cmd -- Returns the text of the most recent Kermit error packet. -- Related: FINISH,KGET,PKT,RECN,RECV,SEND,SERVER
1A873	xKEY	($\rightarrow \text{rc 1}$) ($\rightarrow 0$) Key Cmd -- Returns to level 1 a test result, and if a key is pressed, returns to level 2 the row-column location $xn m$ of that key. -- Related: WAIT,KEYEVAL

21F24	xKGET	<p>(name →) ("name" →) ({names} →) ({{old new}...} →) Kermit Get Cmd --</p>
		<p>Used by a local Kermit to get a Kermit server to transmit the named object(s). --</p>
		<p>Related: BAUD,CKSM,FINISH,PARITY,RECN,RECV,SEND,SERVER,TRANSIO</p>
1A303	xKILL	<p>(→) Cancel Halted Programs Cmd --</p>
		<p>Cancels all currently halted programs. (Halted programs are typically canceled by pressing PRG NXT RUN KILL.) If KILL is executed within a program, that program is also canceled. --</p>
		<p>Related: CONT,DOERR,HALT,PROMPT</p>
1E2D5	xLABEL	<p>(→) Label Axes Cmd --</p>
		<p>Labels axes in PICT with x- and y-axis variable names and with the minimum and maximum values of the display ranges. --</p>
		<p>Related: LABEL,DRAW,DRAX</p>
1A604	xLAST	<p>(→ ob1 .. obn) Last Arguments Cmd --</p>
		<p>Returns copies of the arguments of the most recently executed command. UserRPL: xLASTARG</p>
1E58D	x>LCD	<p>(grob →) Graphics Object to LCD Cmd --</p>
		<p>Displays the graphics object from level 1, with its upper left pixel in the upper left corner of the display. --</p>
		<p>Related: LCD→,BLANK,→GROB UserRPL: x→LCD</p>

1E572	xLCD>	(→ grob) LCD to Graphics Object Cmd -- Returns the current stack and menu display as a 131x64 graphics object. -- Related: →LCD,→GROB UserRPL: xLCD→
2142D	xLIBS	(→ { title nlib nport ... }) Libraries Cmd -- Lists the title, number, and port of each library attached to the current directory. -- Related: ATTACH,DETACH
1E398	xLINE	((x1,y1) (x2,y2) →) ({#n1 #m1} {#n2 #m2} →) Draw Line Cmd -- Draws a line in PICT between the coordinates in levels 1 and 2. -- Related: ARC,BOX,TLINE
200F3	xSIGMALINE	(→ symb) Regression Model Formula Cmd -- Returns an expression representing the best fit line according to the current statistical model, using X as the independent variable name, and explicit values of the slope and intercept taken from the reserved variable ΣPAR. -- <REF>TEXT:Reserved ΣPAR -- Related: BESTFIT,COLΣ,CORR,COV, EXPFIT,LINFIT,LOGFIT,LR,PREDX, PREDY,PWRFIT,XCOL,YCOL UserRPL: xΣLINE
201B1	xLINFIT	(→) Linear Curve Fit Cmd -- Stores LINFIT as the fifth parameter in the reserved variable ΣPAR, indicating that subsequent executions of LR are to use the linear curve fitting model. -- <REF>TEXT:Reserved ΣPAR -- Related: BESTFIT,EXPFIT,LOGFIT,LR, PWRFIT

1C95A	xLIST>	<p>({ } → ob1 . . . obn n) List to Stack Cmd -- Takes a list of n objects and returns them in separate levels, and returns the total number of objects to level 1. -- Related: ARRAY→,DTAG,EQ→,→LIST,OBJ→,STR→ UserRPL: xLIST→</p>
1C783	x>LIST	<p>(ob1 . . . obn n → { }) Stack to List Cmd -- Takes n objects from level n+1 through level 2 and returns a list of those n objects. -- Related: →ARRAY,LIST→,→STR, →TAG,→UNIT UserRPL: x→LIST</p>
1B94F	xLN	<p>(x → x') Natural Logarithm Analytic Func -- Returns the natural (base e) logarithm of the argument. -- z → ln z 'sym' → 'LN(sym)' --</p>
1BA8C	xLNP1	<p>Related: ALOG,EXP,ISOL,LNP1,LOG (x → x') Natural Log of x+1 Analytic Func -- Returns ln (x + 1). -- x → ln(x+1) 'sym' → 'LNP1(sym)' --</p>
1B9C6	xLOG	<p>Related: EXPM,LN (x → x') Common Logarithm Analytic Func -- Returns the common logarithm (base 10) of the argument. -- z → log z 'sym' → 'LOG(sym)' -- Related: ALOG,EXP,ISOL,LN</p>

201D6	xLOGFIT	<p>(\rightarrow) Logarithmic Curve Fit Cmd -- Stores LOGFIT as the fifth parameter in the reserved variable ΣPAR, indicating that subsequent executions of LR are to use the logarithmic curve-fitting model. -- <REF>TEXT:Reserved ΣPAR -- Related: BESTFIT,EXPFIT,LINFIT,LR,PWRFIT</p>
1FF20	xLR	<p>(\rightarrow Intercept Slope) Linear Regression Cmd -- Uses the currently selected statistical model to calculate the linear regression coefficients (intercept and slope) for the selected dependent and independent variables in the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved ΣDAT -- Related: BESTFIT,COLEΣ,CORR,COV,EXPFIT,ΣLINE,LINFIT,LOGFIT,PREDX,PREDY,PWRFIT,XCOL,YCOL</p>
1BE9C	xMANT	<p>($x \rightarrow x'$) Mantissa Func -- Returns the mantissa of the argument. -- $x \rightarrow y_{mant}$ 'sym' \rightarrow 'MANT(sym)' -- Related: SIGN,XPON</p>
1FA8D	xMATCHDN	<p>(symb {spat srepl} \rightarrow symb' 0/1) (symb {spat srepl scond} \rightarrow symb' 0/1) Match Pattern Down Cmd -- Rewrites an expression. -- Related: X\uparrowMATCH UserRPL: x\downarrowMATCH</p>

1FA59	xMATCHUP	<p>(symb {spat srepl} → symb' 0/1) (symb {spat srepl scond} → symb' 0/1) Bottom-Up Match and Replace Cmd -- Rewrites an expression. -- Related: X∇MATCH UserRPL: x↑MATCH</p>
1BC71	xMAX	<p>(x y → x') Maximum Func -- Returns the greater (more positive) of the arguments. -- $x \quad y \quad \rightarrow \max(x, y)$ $x \quad 'sym' \rightarrow 'MAX(x, sym)'$ $'sym' \quad x \quad \rightarrow 'MAX(sym, x)'$ $'sym1' \quad 'sym2' \rightarrow 'MAX(sym1, sym2)'$ $x_u1 \quad y_u2 \rightarrow \max(x_u1, y_u2)$ --</p>
1AADF	xMAXR	<p>Related: MIN (→ MAXR) Maximum Real Func -- Returns the symbolic constant 'MAXR' or its numerical representation, 9.999999999999E499. -- → 'MAXR' → 9.999999999999E499 --</p>
1FE7E	xMAXSIGMA	<p>Related: Ee,i,MINR,π (→ xmax) (→ [x1...xn]) Maximum Sigma Cmd -- Finds the maximum coordinate value in each of the m columns of the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved ΣDAT -- Related: BINS,MEAN,MINΣ,SDEV,TOT,VAR UserRPL: xMAXΣ</p>

1FE99	xMEAN	<p>(\rightarrow xmean) (\rightarrow [x1...xn]) Mean Cmd -- Returns the mean of each of the m columns of coordinate values in the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved ΣDAT --</p>
20FAA	xMEM	<p>Related: BINS,MAXΣ,MINΣ,SDEV,TOT,VAR (\rightarrow x) Memory Available Cmd -- Returns the number of bytes of available RAM. --</p>
21196	xMENU	<p>Related: BYTES (% \rightarrow) Display Menu Cmd -- Displays a built-in menu or a library menu, or displays a custom menu. -- namemenu \rightarrow { listdefinition } \rightarrow 'namedefinition' \rightarrow obj \rightarrow --</p>
2137F	xMERGE	<p>Related: RCLMENU,TMENU (1 \rightarrow) Merge RAM Card Cmd Only useful on the 48. -- Merges the RAM from the card in port 1 with the rest of main user memory. Merged memory is no longer independent. -- Related: FREE,FREE1</p>

1BCE3	xMIN	<p>($x\ y \rightarrow x'$) Minumum Func -- Returns the lesser (more negative) of its two arguments. -- $x\ y \rightarrow \min(x, y)$ $x\ 'sym' \rightarrow 'MIN(x, sym)'$ $'sym'\ x \rightarrow 'MIN(sym, x)'$ $'sym1'\ 'sym2' \rightarrow 'MIN(sym1, sym2)'$ $x_u1\ y_u2 \rightarrow \min(x_u1, y_u2)$ --</p>
1AB01	xMINR	<p>Related: MAX ($\rightarrow MINR$) Minimum Real Func -- Returns the symbolic constant 'MINR' or its numerical representation, 1.000000000000E-499. -- $\rightarrow 'MAXR'$ $\rightarrow 1.000000000000E-499$ --</p>
1FEB4	xMINSIGMA	<p>Related: e,i,MAXR,π ($\rightarrow xmin$) ($\rightarrow [x1\dots xn]$) Minimum Sigma Cmd -- Finds the minimum coordinate value in each of the m current statistics matrix (reserved variable ΣDAT). -- $\langle REF \rangle TEXT:Reserved \Sigma DAT$ --</p>
1BE4D	xMOD	<p>Related: BINS,MAXΣ,MEAN,SDEV,TOT,VAR UserRPL: xMINΣ ($x\ y \rightarrow x'$) Modulo Func -- Returns a remainder defined by: $x \bmod y = x - y \text{ floor}(x/y)$ -- $x\ y \rightarrow x \bmod y$ $x\ 'sym' \rightarrow 'MOD(x, sym)'$ $'sym'\ x \rightarrow 'MOD(sym, x)'$ $'sym1'\ 'sym2' \rightarrow 'MOD(sym1, sym2)'$ -- Related: FLOOR, /</p>

6.3 N-S

1A995	xNEG	<p>($x \rightarrow x'$) Negate Analytic Func -- Changes the sign or negates an object. -- z → -z #n1 → #n2 [arr] → [-arr] 'sym' → '-(sym)' x_u → -x_u grob1 → grob2 PICT1 → PICT2 -- Related: ABS,CONJ,NOT,SIGN</p>
1A2BC	xNEWOB	<p>(ob → ob) New Object Cmd -- Creates a new copy of the specified object. -- Related: MEM,PURGE</p>
2324C	xNEXT	<p>(→) NEXT Cmd -- Ends definite loop structures. See the FOR and START command entries for syntax information. -- Related: FOR,START,STEP</p>
1E88F	xNOT	<p>($x \rightarrow x'$) NOT Cmd -- Returns the one's complement or the logical inverse of the argument. -- #n1 → #n2 T/F → 0/1 "str1" → "str2" 'sym' → 'NOT sym' -- Related: AND,OR,XOR</p>

1FDA6	xNSIGMA	<p>(\rightarrow n rows) Number of Rows Cmd -- Returns the number of rows in the current statistical matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved ΣDAT --</p>
1CB46	xNUM	<p>Related: ΣX,ΣXY,ΣX2,ΣY,ΣY2 UserRPL: xNΣ (\$ \rightarrow n) Character Number Cmd -- Returns the character code n for the first character in the string. --</p>
1A5E4	x>NUM	<p>Related: CHR,POS,REPL,SIZE,SUB (x \rightarrow x') Evaluate to Number Cmd -- Evaluates a symbolic argument object and returns the numerical result. -- objsym \rightarrow z --</p>
1CF7B	xOBJ>	<p>Related: \rightarrowQ,\rightarrowQpi UserRPL: x\rightarrowNUM (ob \rightarrow ?) Object to Stack Cmd -- Separates an object into its components onto the stack. For some object types, the number of composites is returned to level 1. -- (x,y) \rightarrow x y {obj1 ... objn} \rightarrow obj1 objn n [x1 ... xn] \rightarrow x1 xn {n} [[x11 ... xm n]] \rightarrow x11 xm n {m n} "obj" \rightarrow evaluated-obj 'sym' \rightarrow obj1 ... objn n func x_u \rightarrow x 1_u :tag:obj \rightarrow obj "tag" -- Related: ARRAY\rightarrow,C\rightarrowR,DTAG,EQ\rightarrow, R\rightarrowC,STR\rightarrow,\rightarrowTAG UserRPL: xOBJ\rightarrow</p>

1C5AA	xOCT	(→) Octal Mode Cmd -- Selects octal base for binary integer operations. (The default base is decimal.) --
1A31E	xOFF	Related: BIN,DEC,HEX,RCWS,STWS (→) Off Cmd -- Turns off the calculator. -- Related: CONT,HALT,KILL
1EE38	xOLDPRT	Old Printer Cmd -- Modifies the remapping string in the reserved variable PRTPAR so that the extended character set of the HP 48 matches that of the HP 82240A Infrared Printer. Not useful on the 49G.
21EB5	xOPENIO	(→) Open I/O Port Cmd -- Opens the serial port or the IR port using the I/O parameters in the reserved variable IOPAR. -- <REF>TEXT:Reserved IOPAR -- Related: BUFLLEN,CLOSEIO,SBRK,SRECV,STIME,XMIT
1E809	xOR	(x y → x') OR Func -- Returns the logical OR of two arguments. -- #n1 #n2 → #n3 "str1" "str2" → "str3" T/F1 T/F2 → 0/1 T/F 'sym' → 'T/F OR sym' 'sym' T/F → 'sym OR T/F' 'sym1' 'sym2' → 'sym1 OR sym2' -- Related: AND,NOT,XOR

20FD9	xORDER	({names} →) Order Variables Cmd -- Reorders the variables in the current directory (shown in the VAR menu) to the order specified. --
1FC29	xOVER	Related: VARS (1 2 → 1 2 1) Over Cmd -- Returns a copy to stack level 1 of the object in level 2. --
1E6C1	xPARAMETRIC	Related: PICK,ROLL,ROLLD,ROT,SWAP (→) Parametric Plot Type Cmd -- Sets the plot type to PARAMETRIC. --
2202C	xPARITY	Related: BAR,CONTOUR,DIFFEQ,FUNCTION,GRIDMAP,HISTOGRAM,PARSURFACE,PCONTOUR,POLAR,SCATTER,SLOPEFIELD,TRUTH,WIRE-FRAME,YSLICE (n →) Parity Cmd -- Sets the parity value in the reserved variable IOPAR. -- <REF>TEXT:Reserved IOPAR --
1A125	xPATH	Related: BAUD,CKSM,TRANSIO (→ {HOME dir1 .. dirn}) Current Path Cmd -- Returns a list specifying the path to the current directory. --
1E201	xPDIM	Related: CRDIR,HOME,PGDIR,UPDIR ((xmin,ymin) (xmax,ymax) →) (#width #height →) PICT Dimension Cmd -- Replaces PICT with a blank PICT of the specified dimensions. -- Related: PMAX,PMIN

1C236	xPERM	<p>($n\ k \rightarrow n$) Permutations Func -- Returns the number of possible permutations of n items taken m at a time. -- $n\ m \rightarrow P_{n,m}$ 'symn' $m \rightarrow$ 'PERM(symn,m)' n 'symm' \rightarrow 'PERM(n,symm)' 'symn' 'symm' \rightarrow 'PERM(symn,symm)' -- Related: COMB,!</p>
2123A	xPGDIR	<p>(name \rightarrow) Purge Directory Cmd -- Purges the named directory (whether empty or not). -- Related: CLVAR,CRDIR,HOME,PATH,PURGE,UPDIR</p>
1FC9A	xPICK	<p>($1 \dots n\ n \rightarrow 1 \dots n\ 1$) Pick Object Cmd -- Copies the contents of a specified level to level 1. -- Related: DUP,DUPN,DUP2,OVER,ROLL,ROLLD,ROT,SWAP</p>
1E436	xPICT	<p>(\rightarrow PICT) PICT Cmd -- Puts the name PICT on the stack. -- Related: GOR,GCOR,NEG,PICTURE,PVIEW,RCL,REPL,SIZE,STO,SUB</p>
1E2BA	xGRAPH	<p>(\rightarrow) Picture Environment Cmd -- Selects the Picture environment (selects the graphics display and activates the graphics cursor and Picture menu). -- Related: PVIEW,TEXT,PIC UserRPL: xPICTURE</p>

1E36E	xPIX?	((x,y) → 1/0) ({#n #m} → 1/0) Pixel On? Cmd -- Tests whether the specified pixel in PICT is on; returns 1 (true) if the pixel is on, and 0 (false) if the pixel is off. --
1E344	xPIXOFF	Related: PIXON,PIXOFF ((x,y) →) ({#n #m} →) Pixel Off Cmd -- Turns off the pixel at the specified coordinate in PICT. --
1E31A	xPIXON	Related: PIX?,PIXON ((x,y) →) ({#n #m} →) Pixel On Cmd -- Turns on the pixel at the specified coordinate in PICT. --
220DD	xPKT	Related: PIX?,PIXOFF (\$data \$type → \$response) Packet Cmd -- Used to send command "packets" (and receive requested data) to a Kermit server. --
1E09E	xPMAX	Related: CLOSEIO,KERRM,SERVER ((x,y) →) PICT Maximum Cmd -- Specifies (x,y) as the coordinates at the upper right corner of the display. --
1E07E	xPMIN	Related: PDIM,PMIN,XRNG,YRNG ((x,y) →) PICT Minimum Cmd -- Specifies (x,y) as the coordinates at the lower left corner of the display. -- Related: PDIM,PMAX,XRNG,YRNG

1CAB4	xPOS	<p>(str substring \rightarrow n/0) ({} ob \rightarrow n/0) Position Cmd -- Returns the position of a substring within a string or the position of an object within a list. -- Related: CHR,NUM,REPL,SIZE,SUB</p>
1EE53	xPR1	<p>(ob \rightarrow ob) Print Level 1 Cmd -- Prints an object in multiline printer format. -- Related: CR,DELAY,OLDPRT,PRTLCD,PRST, PRSTC,PRVAR</p>
1FF7A	xPREDV	<p>(x \rightarrow y) Predicted y-Value Cmd -- Returns the predicted dependent variable value ydep, based on the independent-variable value xindep, the currently selected stat model, and the current regression coefficients in the reserved variable ΣPAR. -- <REF>TEXT:Reserved ΣPAR --</p>
1FFBA	xPREDX	<p>Related: PREDX (y \rightarrow x) Predicted x-Value Cmd -- Returns the predicted dependent variable value xindepend, based on the independent-variable value ydepend, the currently selected stat model, and the current regression coefficients in the reserved variable ΣPAR. -- <REF>TEXT:Reserved ΣPAR -- Related: COLΣ,CORR,COV,EXPFIT, ΣLINE,LINFIT,LOGFIT,LR, PREDY,PWRFIT,XCOL,YCOL</p>

1FF9A	xPREDY	<p>($x \rightarrow y$) Predicted y-Value Cmd -- Returns the predicted dependent variable value ydepend, based on the independent-variable value xindepend, the currently selected stat model, and the current regression coefficients in the reserved variable ΣPAR. -- <REF>TEXT:Reserved ΣPAR -- Related: $\text{COL}\Sigma, \text{CORR}, \text{COV}, \text{EXPFIT}, \Sigma\text{LINE}, \text{LINFIT}, \text{LOGFIT}, \text{LR}, \text{PREDX}, \text{PWRFIT}, \text{XCOL}, \text{YCOL}$</p>
1EF63	xPRLCD	<p>(\rightarrow) Print LCD Cmd -- Prints a pixel-by-pixel image of the current display (excluding the annunciators) -- Related: $\text{CR}, \text{DELAY}, \text{OLDPRT}, \text{PRST}, \text{PRSTC}, \text{PRVAR}, \text{PR1}$</p>
23824	xPROMPT	<p>($\\$ \rightarrow$) Prompt Cmd -- Displays the contents of "prompt" in the status area, and halts program execution. -- Related: $\text{CONT}, \text{DISP}, \text{FREEZE}, \text{HALT}, \text{INFORM}, \text{INPUT}, \text{MSGBOX}$</p>
1EE89	xPRST	<p>(\rightarrow) Print Stack Cmd -- Prints all objects in the stack, starting with the object in the highest level. -- Related: $\text{CR}, \text{DELAY}, \text{OLDPRT}, \text{PRLCD}, \text{PRSTC}, \text{PRVAR}, \text{PR1}$</p>
1EE6E	xPRSTC	<p>Print Stack (Compact) Cmd -- Prints in compact form all objects in the stack, starting with the object in the highest level. -- Related: $\text{PR}, \text{DELAY}, \text{OLDPRT}, \text{PRLCD}, \text{PRST}, \text{PRVAR}, \text{PR1}$</p>

1EEBF	xPRVAR	<p>(name →) ({names} →) (:port:name →) Print Variable Cmd -- Searches the current directory path or port for the specified variables and prints the name and contents of each variable. -- Related: CP,DELAY,OLDPRT,PR1,PRLCD,PRST,PRSTC</p>
20EFE	xPURGE	<p>(name →) ({names} →) (PICT →) Purge Cmd -- Purges the named variables or empty subdirectories from the current directory. -- Related: CLEAR,CLVAR,NEWOB,PGDIR</p>
1D407	xPUT	<p>(ob pos obj → ob') ob = [] or [[]] or {} or name pos = n or {n} or {n m} Put Element Cmd -- In the level 3 array or list, PUT replaces with zput or objput the object whose position is specified in level 2; if the array or list is unnamed, returns the new array or list. -- Related: GET,GETI,PUTI</p>
1D5DF	xPUTI	<p>(ob pos obj → [] pos') ob = [] or [[]] or {} or name pos = n or {n} or {n m} Put and Increment Index Cmd -- In the level 3 array or list, replaces with zput or objput the object whose position is specified in level 2, returning the new array or list and the next position in that array or list. -- Related: GET,GETI,PUT</p>

211FC	xPVARs	<p>(nport \rightarrow { } mem) Port-Variables Cmd -- Returns a list of the backup objects (:nport:name) and the library objects (:nport:library) in the specified port. Also returns the available memory size (if RAM) or the memory type. -- Related: VARS</p>
1E2F0	xPVIEW	<p>((x, y) \rightarrow) ({ #n #m } \rightarrow) PICT View Cmd -- Displays PICT with the specified coordinate at the upper left corner of the graphics display. -- Related: FREEZE, PICTURE, PICT, TEXT</p>
20220	xPWRFIT	<p>Power Curve Fit Cmd -- Stores PWRFIT as the fifth parameter in the reserved variable ΣPAR, indicating that subsequent executions of LR are to use the power curve fitting model. -- <REF>TEXT:Reserved ΣPAR -- Related: BESTFIT, EXPFIT, LINFIT, LOGFIT, LR</p>
1E27A	xPX>C	<p>({ #m #n } \rightarrow (x, y)) Pixel to Complex Cmd -- Converts the specified pixel coordinates to user-unit coordinates. --</p>
1F9C4	x \rightarrow Q	<p>Related: C\rightarrowPX UserRPL: xPX\rightarrowC (x \rightarrow a/b) To Quotient Cmd -- Returns a rational form of the argument. -- x \rightarrow 'a/b' (x, y) \rightarrow 'a/b+c/d*i' 'sym1' \rightarrow 'sym2' -- Related: \rightarrowQπ, / UserRPL: x\rightarrowQ</p>

1F9E9	x->QPI	<p>($x \rightarrow \text{symp}$) To Quotient Times π Cmd -- Returns a rational form of the argument, or a rational form of the argument with π factored out, whichever yields the smaller denominator. -- $x \rightarrow 'a/b*\pi'$ $x \rightarrow 'a/b'$ $'\text{sym1}' \rightarrow '\text{symp2}'$ $(x,y) \rightarrow 'a/b*\pi+c/d*\pi*i'$ $(x,y) \rightarrow 'a/b+c/d*i'$ --</p>
20AB3	xQUAD	<p>Related: $\rightarrow Q, /, \pi$ UserRPL: $x \rightarrow Q\pi$ ($\text{symp var} \rightarrow \text{symp}'$) Solve Quadratic Equation Cmd -- Solves an algebraic object symp for the variable var, and returns an expression symp' representing the solution. --</p>
1F500	xQUOTE	<p>Related: COLCT,EXPAN,ISOL,SHOW,SOLVE ($ob \rightarrow 'ob$) Quote Argument Func -- Returns its argument unevaluated. -- $'\text{sym}' \rightarrow '\text{sym}'$ $obj \rightarrow obj$ --</p>
1C3B4	xRAD	<p>Related: APPLY, (\rightarrow) Radians Mode Cmd -- Sets Radians angle mode. --</p>
1C1B9	xRAND	<p>Related: DEG,RAD ($\rightarrow x$) Random Number Cmd -- Returns a pseudo-random number generated using a seed value, and updates the seed value. -- Related: COMB,PERM,RDZ,!</p>

1FB5D	xPREDIV	<p>($x\ y \rightarrow x/y$) Prefix Divide Func -- Prefix form of / (divide) generated by the Equation Writer Application. -- z1 z2 \rightarrow z1/z2 [arr] [[mat]] \rightarrow [[arrmat⁻¹]] [arr] z \rightarrow [arr/z] z 'sym' \rightarrow 'z/sym' 'sym' z \rightarrow 'sym/z' 'sym1' 'sym2' \rightarrow 'sym1/sym2' #n1 n2 \rightarrow #n3 n1 #n2 \rightarrow #n3 #n1 #n2 \rightarrow #n3 x_u1 y_u2 \rightarrow (x/y)_u1/u2 x y_u \rightarrow (x/y)_1/u x_u y \rightarrow (x/y)_u 'sym' x_u \rightarrow 'sym/x_u' x_u 'sym' \rightarrow 'x_u/sym' -- Related: / UserRPL: xRATIO</p>
1F133	xRCEQ	<p>(\rightarrow EQ) Recall from EQ Cmd -- Returns the unevaluated contents of the reserved variable EQ from the current directory. -- <REF>TEXT:Reserved EQ -- Related: STEQ</p>
20B40	xRCL	<p>(var \rightarrow x) (:port:nlib \rightarrow lib) (:port:name \rightarrow ob) (:port:{path} \rightarrow ob) Recall Cmd -- Returns the unevaluated contents of a specified variable or plug -in object. --</p>
19928	xRCLALARM	<p>Related: STO (n \rightarrow {date time action rep}) Recall Alarm Cmd -- Recalls a specified alarm. -- Related: DELALARM,FINDALARM,STOALARM</p>

1C619	xRCLF	<p>(\rightarrow {#s1 #u1 #s2 #u2}) Recall Flags Cmd -- Returns a list containing four 64 bit binary integers representing the states of the 64 system and user flags, respectively. --</p>
22586	xRCLKEYS	<p>Related: STOF (\rightarrow {ob ... key ...}) (\rightarrow {S ob ... key ...}) Recall Key Assignments Cmd -- Returns the current user key assignments. This includes an S if the standard key definitions are active (not suppressed) for those keys without user key assignments. --</p>
211E1	xRCLMENU	<p>Related: ASN,DELKEYS,STOKEYS (\rightarrow x) Recall Menu Number Cmd -- Returns the menu number of the currently displayed menu. --</p>
1FD46	xRCLSIGMA	<p>Related: MENU,TMENU (\rightarrow [[]]) Recall Sigma Cmd -- Returns the current stat matrix (the contents of reserved var ΣDAT) from the current directory. -- <REF>TEXT:Reserved ΣDAT --</p>
1C5FE	xRCWS	<p>Related: CLΣ,STOΣ,Σ+,Σ- UserRPL: xRCLΣ (\rightarrow n) Recall Wordsize Cmd -- Returns the current wordsize in bits (1 through 64). -- Related: BIN,DEC,HEX,OCT,STWS</p>

1D0DF	xRDM	<p>(ob size \rightarrow ob') (name size \rightarrow) ob= [] or [[]] size = {n} or {n m} Redimension Array Cmd -- Rearranges the elements of the argument according to the specified dimensions. --</p>
1C1D4	xRDZ	<p>Related: TRN (x \rightarrow) Randomize Cmd -- Uses a real number xseed as a seed for the RAND command. --</p>
1C7CA	xRE	<p>Related: COMB,PERM,RAND,! ((x,y) \rightarrow x) ([] \rightarrow []') Real Part Func -- Returns the real part of the argument. --</p> <p>x \rightarrow x x_u \rightarrow x (x,y) \rightarrow x [R-arr] \rightarrow [R-arr] [C-arr] \rightarrow [R-arr] 'sym' \rightarrow 'RE(sym)' --</p>
21F62	xRECN	<p>Related: C\rightarrowR,IM,R\rightarrowC (name \rightarrow) (\$name \rightarrow) Receive Renamed Object Cmd -- Prepares the HP 48 to receive a file from another Kermit device, and to store the file in a specified variable. --</p> <p>Related: BAUD,CKSM,CLOSEIO,FINISH, KERRM,KGET,PARITY,RECV,SEND, SERVER,TRANSIO</p>

21F96	xRECV	<p>(→) Receive Object Cmd -- Instructs the HP 48 to look for a named file from another Kermit device. The received file is stored in a variable named by the sender. -- Related: BAUD,CKSM,FINISH,KGET,PARITY,RECN,SEND,SERVER,TRANSIO</p>
2305D	xREPEAT	<p>(1/0 →) REPEAT Cmd -- Starts loop clause in WHILE ... REPEAT ... END indefinite loop structure. -- Related: END,WHILE</p>
1C8EA	xREPL	<p>(ob pos new → ob') ob= [[]] or [] or {} or \$ or PICT pos= N or {n m} or (n,m) Replace Cmd -- Replaces a portion of the level 3 target object with the level 1 object, beginning at a position specified in level 2. --</p>
1E126	xRES	<p>Related: CHR,GOR,GXOR,NUM,POS,SIZE,SUB (n_int →) (#n_int →) Resolution Cmd -- Specifies the resolution of mathematical and statistical plots, where the resolution is the interval between values of the independent variable used to generate the plots. -- Related: BAR,CONIC,DIFFEQ,FUNCTION,GRIDMAP,HISTOGRAM,PARAMETRIC,PARSURFACE,PCONTOUR,POLAR,SCATTER,SLOPEFIELD,TRUTH,WIREFRAME,YSLICE</p>

2133C	xRESTORE	<pre>(:port:name →) (ob →) Restore HOME Cmd -- Replaces the current HOME directory with the specified backup copy. -- :nport:namebackup → obj backup → --</pre>
1959B	xRL	<pre>Related: ARCHIVE (# → #') Rotate Left Cmd -- Rotates a binary integer one bit to the left. --</pre>
195BB	xRLB	<pre>Related: RLB,RR,RRB (# → #') Rotate Left Byte Cmd -- Rotates a binary integer one byte to the left. --</pre>
1BD55	xRND	<pre>Related: RL,RR,RRB (x n → x') Round Func -- Rounds an object to a specified number of decimal places or significant digits, or to fit the current display format. -- z1 nrnd → z2 z 'symrnd' → 'RND(z,symrnd)' 'sym' nrnd → 'RND(symb,nrnd)' 'sym1' 'symrnd' → 'RND(sym1,symrnd)' [arr1] nrnd → [arr2] x_u nrnd → y_u x_u 'symrnd' → 'RND(x_u,symrnd)' --</pre>
1BF9E	xRNRM	<pre>Related: TRNC ([] → x) Row Norm Cmd -- Returns the row norm (infinity norm) of its argument array. -- Related: CNRM,CROSS,DET,DOT</pre>

1FCB5	xROLL	$(1 \dots n \ n \rightarrow 2 \dots n \ 1)$ Roll Objects Cmd -- Moves the contents of a specified level to level 1, and rolls upwards the portion of the stack beneath the specified level. --
1FCD0	xROLLD	Related: OVER,PICK,ROLLD,ROT,SWAP $(n \dots 1 \ n \rightarrow 1 \ n \dots 2)$ Roll Down Cmd -- Moves the contents of level 1 to a specified level, and rolls downwards the portion of the stack beneath the specified level --
1F16E	xROOT	Related: OVER,PICK,ROLL,ROT,SWAP $(\text{prog/s var guess} \rightarrow x)$ $(\text{prog/s var \{guesses\}} \rightarrow x)$ Root-Finder Cmd -- Returns a real number xroot that is a value of the specified variable var for which the specified program or algebraic object most nearly evaluates to zero or a local extremum.
1FC0E	xROT	$(1 \ 2 \ 3 \rightarrow 2 \ 3 \ 1)$ Rotate Objects Cmd -- Rotates the first three objects on the stack, moving the object in level 3 to level 1. --
195DB	xRR	Related: OVER,PICK,ROLL,ROLLD,SWAP,UNROT $(\# \rightarrow x')$ Rotate Right Cmd -- Rotates a binary integer one bit to the right. --
195FB	xRRB	Related: RL,RLB,RRB $(\# \rightarrow x')$ Rotate Right Byte Cmd -- Rotates a binary integer one byte to the right. -- Related: RL,RLB,RR

1C03E	xRSD	([B] [[A]] [Z] → [] ') ([[B]] [[A]] [[Z]] → [[]] ') Residual Cmd -- Computes the residual $B - AZ$ of the arrays B, A, and Z.
20A7D	xRULES	
1969B	xR>B	(x → #) Real to Binary Cmd -- Converts a positive real integer to its binary integer equivalent. -- Related: B→R UserRPL: xR→B
1C79E	xR>C	(x y → (x,y)) ([X] [Y] → [(x,y)]) Real to Complex Cmd -- Combines two real numbers or real arrays into a single complex number or array. -- Related: C→R,IM,RE UserRPL: xR→C
1BEF4	xR>D	(x → (180/π)x) Radians to Degrees Func -- Converts a real number expressed in radians to its equivalent in degrees. -- x → (180/π)x 'sym' → 'R→D(sym)' -- Related: D→R UserRPL: xR→D
1E761	xSAME	(ob1 ob2 → 1/0) Display information about the makers of the calculator. Same Object Cmd -- Compares two objects, and returns a true result (1) if they are identical, and a false result (0) if they are not. --
220C2	xSBRK	Related: TYPE,== (→) Serial Break Cmd -- Interrupts serial transmission or reception. -- Related: BUFLen,SRECV,STIME,XMIT

1E1E1	xSCALE	(<i>xs ys</i> →) Scale Plot Cmd -- Adjusts the first two parameters in PPAR, (<i>xmin</i> , <i>ymin</i>) and (<i>xmax</i> , <i>ymax</i>), so that <i>xscale</i> and <i>yscale</i> are the new plot horizontal and vertical scales, and the center point doesn't change. -- <REF>TEXT:Reserved PPAR --
1E150	x*H	Related: AUTO,CENTR,SCALEH,SCALEW (<i>xf</i> →) Multiply Height Cmd -- Multiplies the vertical plot scale by <i>xfactor</i> . --
1E170	x*W	Related: AUTO,SCALEW,YRING UserRPL: xSCALEH (<i>yf</i> →) Multiply Width Cmd -- Multiplies a plot's horizontal scale by <i>xfactor</i> . --
2018C	xSCATRLOT	Related: AUTO,SCALEH,YRING UserRPL: xSCALEW (→) Draw Scatter Plot Cmd -- Draws a scatter plot of (<i>x</i> , <i>y</i>) data points from the specified columns of the current statistics matrix (reserved variable Σ DAT). --
1E701	xSCATTER	Related: BARPLOT,PICTURE,HISTPLOT, PVIEW,SCLΣ,XCOL,YCOL Scatter Plot Type Cmd -- Sets the plot type to SCATTER. -- Related: BAR,CONIC,DIFFEQ,FUNCTION, GRIDMAP,HISTOGRAM,PARAMETRIC,PARSURFACE, PCONTOUR,POLAR,SLOPEFIELD,TRUTH, WIRE- FRAME,YSLICE

1C41E	xSCI	($n \rightarrow$) Scientific Mode Cmd -- Sets the number display format to Scientific mode, which displays one digit to the left of the fraction mark and n significant digits to the right. --
200C4	xSCLSIGMA	Related: ENG, FIX, STD (\rightarrow) Scale Sigma Cmd -- Adjusts (xmin,ymin) and (xmax, ymax) in PPAR so that a subsequent scatter plot exactly fills PICT. -- <REF>TEXT:Reserved PPAR --
203CC	xSCONJ	Related: AUTO, SCATRLOT UserRPL: xSCLΣ ($name \rightarrow$) Store Conjugate Cmd -- Conjugates the contents of a named object. --
1FECE	xSDEV	Related: CONJ, SIN, SNEG (\rightarrow xsdev) (\rightarrow [x1...xn]) Standard Deviation Cmd -- Calculates the sample standard deviation of each of the m columns of coordinate values in the current stat matrix (reserved var ΣDAT). --
21EF0	xSEND	Related: MAXΣ, MEAN, MINΣ, PSDEV, PVAR, TOT, VAR ($name \rightarrow$) ({names} \rightarrow) ({{old new}...} \rightarrow) Send Object Cmd -- Sends a copy of the named object to a Kermit device. -- Related: BAUD, CLOSEIO, CKSM, FINISH, KERRM, KGET, PARITY, RECN, RECV, SERVER, TRANSIO

21FD1	xSERVER	(\rightarrow) Server Mode Cmd -- Selects Kermit Server mode. -- Related: BAUD,CKSM,FINISH,KERRM, KGET,PARITY,PKT,RECN,RECV, SEND,TRANSIO
1C274	xSF	($n \rightarrow$) Set Flag Cmd -- Sets a specified user or system flag. -- Related: CF,FC?,FC?C,FS?,FS?C
20AD3	xSHOW	(<i>syb name</i> \rightarrow <i>syb'</i>) (<i>syb {names}</i> \rightarrow <i>syb'</i>) Show Variable Cmd -- Returns <i>syb'</i> which is equivalent to <i>syb</i> except that all implicit references to a variable name are made explicit. -- Related: COLCT,EXPAN,ISOL,QUAD
1B32A	xSIGN	($x \rightarrow x'$) Sign Func -- Returns the sign of a real number argument, the sign of the numerical part of a unit object argument, or the unit vector in the direction of a complex number argument. -- Related: ABS,MANT,XPON
1B4AC	xSIN	($x \rightarrow x'$) Sine Analytic Func -- $z \rightarrow \sin z$ ' <i>sym</i> ' \rightarrow 'SIN(<i>sym</i>)' <i>x_uangular</i> $\rightarrow \sin(x_uangular)$ -- Related: ASIN,COS,TAN

1B5B7	xSINH	<p>($x \rightarrow x'$) Hyperbolic Sine Analytic Func -- Returns the hyperbolic sine of the argument. -- $z \rightarrow \sinh z$ 'sym' \rightarrow 'SINH(sym)' -- Related: ANUSH,COSH,TANH</p>
202CE	xSINV	<p>(name \rightarrow) Store Inverse Cmd -- Replaces the contents of the named variable with its inverse. -- Related: INV,SCONJ,SNEG</p>
1C9B8	xSIZE	<p>(ob \rightarrow n) (ob \rightarrow {N m}) (ob \rightarrow #nw #nh) Size Cmd -- Returns the number of characters in a string, the number of elements in a list, the dimensions of an array, the number of objects in a unit object or algebraic object, or the dimensions of a graphics object. -- "str" \rightarrow n { list } \rightarrow n [vector] \rightarrow { n } [[mat]] \rightarrow { n m } 'sym' \rightarrow n grob \rightarrow #nwidth #mheight PICT \rightarrow #nwidth #mheight x_u \rightarrow n -- Related: CHR,NUM,POS,REPL,SUB</p>
1961B	xSL	<p>(# \rightarrow #') Shift Left Cmd -- Shifts a binary integer one bit to the left. -- Related: ASR,SLB,SR,SRB</p>

1963B	xSLB	(# → #') Shift Left Byte Cmd -- Shifts a binary integer one byte to the left. --
2034D	xSNEG	Related: ASR,SL,SR,SRB (name →) Store Negate Cmd -- Replaces the contents of a variable with its negative. --
1B426	xSQ	Related: NEG,SCONJ,SINV (x → x') Square Analytic Func -- Returns the square of the argument. -- z → z2 x_u → x2_u2 [[mat]] → [[mat mat]] 'sym' → 'SQ(sym)' --
1965B	xSR	Related: \sqrt{a} , [^] (# → #') Shift Right Cmd -- Shifts a binary integer one bit to the right. --
1967B	xSRB	Related: ASR,SL,SLB,SRB (# → #') Shift Right Byte Cmd -- Shifts a binary integer one byte to the right. --
21E95	xSRECV	Related: ASR,SL,SLB,SR (n → \$ 0/1) Serial Receive Cmd -- Reads up to n characters from the serial input buffer and returns them as a string, along with a digit indicating whether errors occurred. -- Related: BUFFLEN,CLOSEIO,OPENIO, SBRK,STIME,XMIT

23103	xSTART	<p>(<i>start finish</i> →) START Definite Loop Structure Cmd -- START <i>xstart xfinish</i> → NEXT <i>xstart xfinish</i> → STEP <i>xincrement</i> → STEP 'symbincrement' → -- Related: FOR,NEXT,STEP</p>
1C486	xSTD	<p>(→) Standard Mode Cmd -- Sets the number display format to Standard mode. -- Related: ENG,FIX,SCI</p>
23380	xSTEP	<p>(<i>n</i> →) (<i>symb</i> →) STEP Cmd -- Defines the increment (step) value, and ends definite loop struct See the FOR and START command entries for syntax information. -- Related: FOR,BEXT,START</p>
1F14E	xSTEQ	<p>(<i>ob</i> →) Store in EQ Cmd -- Stores an object into the reserved variable EQ in the current directory. -- <REF>TEXT:Reserved EQ --</p>
220A2	xSTIME	<p>Related: RCEQ (<i>x/0</i> →) Serial Time-Out Cmd -- Specifies the period that SRECV (serial reception) and XMIT (serial transmission) wait before timing out. -- Related: BU- FLEN,CLOSEIO,SBRK,SRECV,XMIT</p>

20CCD	xSTO	<pre>(ob name →) (ob :port:name →) (lib port →) (bup port →) (ob 'name(i)' →) Store Cmd -- Stores an object into a specified variable or object. --</pre>
20538	xSTO-	<pre>Related: DEFINE,RCL,→ (ob name →) (name ob →) Store Minus Cmd -- Calculates the difference between a number (or other object) and the contents of a specified variable, and stores the new value to the specified variable. --</pre>
20753	xSTO*	<pre>Related: STO+,STO*,STO/,- (ob name →) (name ob →) Store Times Cmd -- Multiplies the contents of a specified variable by a number or other object. --</pre>
2060C	xSTO/	<pre>Related: STO+,STO-,STO/,* (ob name →) (name ob →) Store Divide Cmd -- Calculates the quotient of a number (or other object) and the contents of a specified variable, and stores the new value to the specified variable. --</pre>
2044B	xSTO+	<pre>Related: STO+,STO-,STO*,/ (ob name →) (name ob →) Store Plus Cmd -- Adds a number or other object to the contents of a specified variable. -- Related: STO-,STO*,STO/,+</pre>

198FE	xSTOALARM	(time → n) ({date time act rep} → n) Store Alarm Cmd -- Stores an alarm in the system alarm list and returns its alarm index number. act and rep arguments are optional. --
1C67F	xSTOF	Related: DELALARM,FINDALARM,RCLALARM ({#s1 #u1 #s2 #u2} →) Store Flags Cmd -- Sets the states of the system flags or the system and user flags. --
22514	xSTOKEYS	Related: RCLF,STWS,RCWS ({ob key ...} →) ({ 'S' ob key ... } →) ('S' →) Store Key Assignments Cmd -- Defines multiple keys on the user keyboard by assigning objects to specified keys. --
1FD0B	xSTOSIGMA	Related: ASN,DELKEYS,RCLKEYS (ob →) Store Sigma Cmd -- Stores obj in the reserved variable ΣDAT. --
1CB26	xSTR>	Related: CLΣ,RCLΣ,Σ+,Σ- UserRPL: xSTOΣ (\$ → ob) Evaluate String Cmd -- Evaluates the text of a string as if the text were entered from the command line. --
1CB0B	x>STR	Related: ARRAY→,DTAG,EQ→,LIST→, OBJ→,→STR UserRPL: xSTR→ (ob → \$) Object to String Cmd -- Converts any object to string form. -- Related: →ARRAY,→LIST,STR→, →TAG,→UNIT UserRPL: x→STR

1C5C5	xSTWS	<p>(n →) (#n →) Set Wordsize Cmd -- Sets the current binary integer wordsize to n bits, where n is a value from 1 through 64 (the default is 64). -- Related: BIN,DEC,HEX,OCT,RCWS</p>
1C85C	xSUB	<p>(ob start end → ob') ob= [[]], \$, {}, grob, PICT start,end = n, {n m}, (n,m) Subset Cmd -- Returns the portion of a string or list defined by specified positions, or returns the rectangular portion of a graphics object or PICT defined by two corner pixel coordinates. -- Related: CHR,GOR,GXOR,NUM,POS,REPL,SIZE</p>
1FBBD	xSWAP	<p>(ob1 ob2 → ob2 ob1) Swap Objects Cmd -- Interchanges the first two objects on the stack. -- Related: DUP,DUPN,DUP2,OVER,PICK,ROLL,ROLLD,ROT</p>
1A52E	xSYSEVAL	<p>(# → ?) Evaluate System Object Cmd -- Evaluates unnamed operating system objects specified by their memory addresses. -- Related: EVAL,LIBEVAL,FLASHEVAL</p>

6.4 T-Z

1C0D7	x%T	<p>($x y \rightarrow 100y/x$) Percent of Total Function -- Returns the percent of the level 2 argument that is represented by the level 1 argument. -- $x \quad y \quad \rightarrow 100y/x$ $x \quad 'sym' \rightarrow '%T(x,sym)'$ $'sym' \quad x \quad \rightarrow '%T(sym,x)'$ $'sym1' \quad 'sym2' \rightarrow '%T(sym1,sym2)'$ $x_u1 \quad y_u2 \rightarrow 100y_u2/x_u1$ $x_u \quad 'sym' \rightarrow '%T(x_u,sym)'$ $'sym' \quad x_u \quad \rightarrow '%T(sym,x_u)'$ -- Related: %, %ch</p>
225BE	x->TAG	<p>($ob \ tag \rightarrow :tag:ob$) Stack to Tag Cmd -- Combines objects in levels 1 and 2 to create a tagged (labeled) object. Tag may be any object. It will be converted to a string. -- Related: \rightarrowARRY,DTAG,\rightarrowLIST,OBJ\rightarrow, \rightarrowSTR,\rightarrowUNIT UserRPL: x\rightarrowTAG</p>
1B55E	xTAN	<p>($x \rightarrow x'$) Tangent Analytic Func -- Returns the tangent of the argument. -- $z \quad \rightarrow \tan z$ $'sym' \quad \rightarrow 'TAN(sym)'$ $x_unitang \rightarrow \tan(x_unitang)$ -- Related: ATAN,COS,SIN</p>
1B655	xTANH	<p>($x \rightarrow x'$) Hyperbolic Tangent Analytic Func -- Returns the hyperbolic tangent of the argument. -- $z \quad \rightarrow \tanh z$ $'sym' \rightarrow 'TANH(sym)'$ -- Related: ATANH,COSH,SINH</p>

20B20	xTAYLR	(symb var n \rightarrow symb') Taylor's Polynomial Cmd -- Calculates the nth order Taylor's polynomial of 'symb' in the variable var. -- Related: ∂, f, Σ
1E606	xTEXT	(\rightarrow) Show Stack Display Cmd -- Displays the stack display. -- Related: PICTURE,PVIEW
22EFA	xTHEN	(0/1 \rightarrow) THEN Cmd -- Starts the true-clause in conditional or error-trapping structure -- Related: CASE,ELSE,END,IFERR
237A8	xTHENCASE	THEN in a CASE statement. -- Related: CASE,ELSE,END,IFERR UserRPL: xTHEN
2371F	xERRTHEN	THEN in an ON ERROR construct. -- Related: CASE,ELSE,END,IFERR UserRPL: xTHEN
1982D	xTICKS	(\rightarrow #) Ticks Cmd -- Returns the system time as a binary integer, in units of 1/8192 second. -- Related: TIME
197F7	xTIME	(\rightarrow time) Time Cmd -- Returns the system time in the form HH.MMSSs. -- Related: DATE,TICKS,TSTR

198BE	xSETTIME	(<i>time</i> →) Set System Time Cmd -- Sets the system time. --
		Related: CLKADJ,→DATE UserRPL: x→TIME
1E3C2	xTLINE	((<i>x1</i> , <i>y1</i>) (<i>x2</i> , <i>y2</i>) →) ({ # <i>n1</i> # <i>m1</i> } { # <i>n2</i> # <i>m2</i> } →) Toggle Line Cmd -- For each pixel along the line in PICT defined by the specified coordinates, TLINE turns off every pixel that is on, and turns on every pixel that is off. --
		Related: ARC,BOX,LINE
2115D	xTMENU	(% → [InitMenu%]) ({ } →) (<i>name</i> →) (0 <i>b</i> → [@LIST InitMenu]) Temporary Menu Cmd -- Displays a built-in menu, library menu, or a user-defined menu. --
		Related: MENU,RCLMENU
1FEAA	xTOT	(→ <i>xsum</i>) (→ { <i>x1</i> . . . <i>xn</i> }) Total Cmd -- Computes the sum of each of the <i>m</i> columns of coordinate values in the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved ΣDAT --
		Related: MAXΣ,MINΣ,MEANMPSDEV, PVAR,SDEV,VAR
2204C	xTRANSIO	(<i>n</i> →) I/O Translation Cmd -- Specifies the character translation option. These translations affect only ASCII Kermit transfer and files printed to the serial port. -- Related: BAUD,CKSM,PARITY

1D392	xTRN	<p>([[]] → [[]] ') (name →) Transpose Matrix Cmd -- Returns the (conjugate) transpose of a matrix. --</p>
1BDD1	xTRNC	<p>Related: CONJ (x n →) Truncate Func -- Truncates an object to a specified number of decimal places or significant digits, or to fit the current display format. -- z1 ntrnc → z2 z1 'symtrnc' → 'TRNC(z1,symtrnc)' 'sym1' ntrnc → 'TRNC(sym1,ntrnc)' 'sym1' 'symtrnc' → 'TRNC(sym1,symtrnc)' [arr]1 ntrnc → [arr]2 x_u ntrnc → y_u x_u 'symtrnc' → 'TRNC(x_u,symtrnc)' --</p>
1E6E1	xTRUTH	<p>Related: RND (→) Truth Plot Type Cmd -- Sets the plot type to TRUTH. -- Related: BAR,CONIC,DIFFEQ,FUNCTION, GRIDMAP,HISTOGRAM,PARAMETRIC,PARSURFACE, PCONTOUR,POLAR,SCATTER,SLOPEFIELD,WIREFRAME,YSLIC</p>
19992	xTSTR	<p>(date time → \$) Date and Time String Cmd -- Returns a string derived from the date and time. -- Related: DATE,TICKS,TIME</p>

```
1A1AF      xTVARS      ( ntype → {} )
              ( {n...} → {} )
              Typed Variables Cmd
              --
              Lists all global variables in the current directory that
              contain objects of the specified types.
              --
              Related: PVAR,TYPE,VAR
```

```

1CB86      xTYPE      ( ob → %type )
Type Cmd
--
Returns the type number of an object.
--
User Objects:
--
Object      Type      Number
-----      ----      -
Real        number     0
Complex     number     1
Character   string     2
Real        Array     3
Complex     Array     4
List        5
Global     name     6
Local      name     7
Program    8
Algebraic  Object    9
Binary     Integer   10
Graphics   object    11
Tagged     object    12
Unit       object    13
XLIB       name     14
Directory  15
Library    16
Backup     object    17
--
Built-in Cmds:
--
Object      Type      Number
-----      ----      -
Built-in    function   18
Built-in    command   19
--
System Objects:
--
Object      Type      Number
-----      ----      -
System      binary    20
Extended    real     21
Extended    complex   22
Linked      array     23
Character   24
Code        object    25
Library     data     26
External    object    26-31
--
Related: SAME, TVARS, VTYPE

```

19771	xUBASE	<p>(u → u') Convert to SI Base Units Func -- Converts a unit object to SI base units. -- x_u → y_base-units 'sym' → 'UBASE(symb)' -- Related: CONVERT,UFACT,→UNIT,UVAL</p>
197A5	xUFACT	<p>(u1 u2 → u3) Factor Unit Cmd -- Factors the level 1 unit from the unit expression of the level 2 unit object. -- Related: CONVERT,UBASE,→UNIT,UVAL</p>
1974F	x>UNIT	<p>(x u → u') Stack to Unit Object Cmd -- Creates a unit object from a real number and the unit part of a unit object. -- Related: →ARRAY,→LIST,→STR,→TAG UserRPL: x→UNIT</p>
230ED	xUNTIL	<p>(→) UNTIL Cmd -- Starts test-clause in DO ... UNTIL ... END indefinite loop structure. -- See the DO entry for syntax info. --</p>
1A15B	xUPDIR	<p>Related: DO,END (→) Up Directory Cmd -- Makes the parent of the current directory the new current directory. -- Related: CRDIR,HOME,PATH,PGDIR</p>

2001A	xUTPC	<p>($n \ x \rightarrow x'$) Upper Chi-Square Distribution Cmd -- Returns the probability $utpc(n,x)$ that a chi-square random variable is greater than x, where n is the number of degrees of freedom of the distribution. --</p>
2005A	xUTPF	<p>Related: UTPF,UTPN,UTPT ($n1 \ n2 \ x \rightarrow x'$) Upper Snedecor's F Distrib. Cmd -- Returns the probability $utpf(n1,n2,x)$ that a Snedecor's F random variable is greater than x, where $n1$ and $n2$ are the numerator and denominator degrees of freedom of the F distribution. --</p>
2003A	xUTPN	<p>Related: UTPC,UTPN,UTPT ($n \ v \ x \rightarrow x'$) Upper Normal Distribution Cmd -- Returns the probability $utpn(m,v,x)$ that a normal random variable is greater than x, where m and v are the mean and variance, respectively, of the normal distribution. --</p>
2007A	xUTPT	<p>Related: UTPC,UTPF,UTPT ($n \ x \rightarrow x'$) Upper Student's t Distrib. Cmd -- Returns the probability $utpt(n,x)$ that a Student's t random variable is greater than x, where n is the number of degrees of freedom of the distribution. --</p>
1971B	xUVAL	<p>Related: UTPC,UTPF,UTPN ($u \rightarrow x$) Unit Value Func -- Returns the numerical part of a unit object. -- $x_u \rightarrow x$ 'sym' \rightarrow 'UVAL(sym)' -- Related: CONVERT,UBASE,UFACT,\rightarrowUNIT</p>

1DD06	xV>	<p>([] / () → x y) ([] / () → x y z) (in current co-system) Vector/Complex Num to Stack Cmd -- [x y] → x y [xr ANgyθ] → xr yθ [x1 x2 x3] → x1 x2 x3 [x1 ANgxθ xz] → x1 xθ xz [x1 ANgxθ ANgx] → x1 xθ x [x1 x2 ... xn] → x1 ... xn (x,y) → x y (xr ANgyθ) → xr yθ --</p>
1DE66	x>V2	<p>Related: →V2, →V3 UserRPL: xV→ (x y → []) (x y → ()) Stack to Vector/Complex Num Cmd -- Converts two numbers from the stack into a 2- element vector or complex number. --</p>
1DEC2	x>V3	<p>Related: V→, →V3 UserRPL: x→V2 (x y z → []) Stack to 3-Element Vector Cmd -- Converts three numbers into a 3-element vector. --</p>
1FF05	xVAR	<p>Related: V→, →V2 UserRPL: x→V3 (→ x) (→ [x1...xn]) Variance Cmd -- Calculates the sample variance of the coordinate val- ues in each of the m columns in the current stat ma- trix (ΣDAT). --</p>
1A194	xVARS	<p>Related: MAXΣ, MEAN, MINΣ, PSDEV, PVAR, SDEV, TOT (→ { }) Variables Cmd -- Returns a list of all variables' names in the VAR menu (the current directory). -- Related: ORDER, PVAR, TVARS</p>

1CE28	xVTYPE	(name \rightarrow n) Variable Type Cmd -- Returns the type number of the object contained in the named variable. -- 'name' \rightarrow ntype :nport:namebackup \rightarrow ntype :nport:nlibrary \rightarrow ntype --
1A71F	xWAIT	Related: TYPE (sec \rightarrow) (0 \rightarrow rc.p) Wait Cmd -- Suspends program execution for specified time, or until a key is pressed. --
23033	xWHILE	Related: KEY (\rightarrow) WHILE Indefinite Loop Struct Cmd -- Starts the WHILE ... REPEAT ... END indefinite loop structure. --
19848	xWSLOG	Related: DO,END,REPEAT (\rightarrow \$ \$ \$ \$) Warmstart Log Cmd -- Returns four strings recording the date, time, and cause of the four most recent warmstart events
1FE2D	xSUMX2	(\rightarrow xsum) Sum of Squares of x-Values Cmd -- Sums the squares of the values in the independent-variable column of the current stat matrix (reserved variable Σ DAT). -- <REF>TEXT:Reserved Σ DAT -- Related: N Σ ,XCOL, Σ X, Σ XY, Σ X2, Σ Y, Σ Y2 UserRPL: x Σ X2

1FFDA	xXCQL	<p>(n →) Independent Column Cmd -- Specifies the independent variable column of the current stat matrix (reserved variable ΣDAT). --</p>
21E75	xXMIT	<p><REF>TEXT:Reserved ΣDAT (\$ → 1) (\$ → \$rest 0) Serial Transmit Cmd -- Sends a string serially without using Kermit protocol, and returns a single digit that indicates whether the transmission was successful. --</p>
1E8F6	xXOR	<p>Related: BUFLN,SBRK,SRECV,STIME (# #' → #') (\$ \$' → \$') (1/0 1/0 → 1/0) Exclusive OR Cmd -- Returns the logical exclusive OR of two arguments. #n1 #n2 → #n3 "str1" "str2" → "str3" T/F1 T/F2 → 0/1 T/F 'sym' → 'T/F XOR sym' 'sym' T/F → 'sym XOR T/F' 'sym1' 'sym2' → 'sym1 XOR sym2' --</p>
1BC45	xXPON	<p>Related: AND,OR,NOT (% → n) Exponent Func -- Returns the exponent of the arg. --</p>
1E621	xXRNG	<p>Related: MANT,SIGN (x1 x2 →) x-Axis Display Range Cmd -- Specifies the x-axis display range. -- Related: AUTO,PDIM,PMAX,PMIN,YRNG</p>

1B1CA	xXROOT	<p>($y \ x \rightarrow Y'$) xth Root of y Cmd -- Computes the xth root of a real number. $y \quad x \quad \rightarrow \ x \ \text{ROOT} \ y$ 'sym1' 'sym2' \rightarrow '$\text{XROOT}(\text{sym2}, \text{sym1})$' '$\text{sym}$' $x \quad \rightarrow$ '$\text{XROOT}(x, \text{sym})$' $y \quad \text{'sym'}$ \rightarrow '$\text{XROOT}(\text{sym}, y)$' $y_u \quad x \quad \rightarrow \ x \ \text{ROOT} \ y_u/x$ $y_u \quad \text{'sym'}$ \rightarrow '$\text{XROOT}(\text{sym}, y_u)$'</p>
1FE63	xSUMXY	<p>($\rightarrow \text{xsum}$) Sum of x Times Y Cmd -- Sums the products of the corresponding values in the independent and dependent variable columns of the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved ΣDAT --</p>
1FE12	xSUMY	<p>Related: $N\Sigma, \text{XC0L}, \Sigma\text{X}, \Sigma\text{X2}, \Sigma\text{Y}, \Sigma\text{Y2}$ UserRPL: $\text{x}\Sigma\text{Y}$ ($\rightarrow \text{xsum}$) Sum of y-Values Cmd -- Sums the values in the dependent variable column of the current stat matrix (reserved var ΣDAT). -- <REF>TEXT:Reserved ΣDAT --</p>
1FE48	xSUMY2	<p>Related: $N\Sigma, \text{XC0L}, \Sigma\text{X}, \Sigma\text{XY}, \Sigma\text{X2}, \Sigma\text{Y2}$ UserRPL: $\text{x}\Sigma\text{Y}$ ($\rightarrow \text{xsum}$) Sum of Squares of y-Values Cmd -- Sums the squares of the values in the dependent-variable column of the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved ΣDAT -- Related: $N\Sigma, \text{XC0L}, \Sigma\text{X}, \Sigma\text{XY}, \Sigma\text{X2}, \Sigma\text{Y}$ UserRPL: $\text{x}\Sigma\text{Y2}$</p>

1FFFA	xYCOL	<p>($n \rightarrow$) Dependent Column Cmd -- Specifies the dependent-variable column of the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved ΣDAT -- Related: BARPLOT,BESTFIT,COLEΣ,CORR,COV,EXPFIT,HISTPLOT,LINFIT,LOGFIT,LR,PREDX,PREFY,PWRFIT,SCATRPLOT,XCOL</p>
1E641	xYRNG	<p>($y_1 y_2 \rightarrow$) y-Axis Display Range Cmd -- Specifies the y-axis display range. -- Related: AUTO,PDIM,PMAX,PMIN,XRNG</p>

6.5 Non A-Z

1B02D	x [^]	<p>($y x \rightarrow y^x$) Power Analytic Func -- Returns the value of the level 2 object raised to the power of the level 1 object. $w \quad z \quad \rightarrow w^z$ $z \quad 'sym' \rightarrow 'z^{sym}'$ $'sym' \quad z \rightarrow '(sym)^z$ $'sym1' \quad 'sym2' \rightarrow 'sym1^{(sym2)}$ $x_u \quad y \rightarrow xy_{uy}$ $x_u \quad 'sym' \rightarrow '(x_u)^{(sym)}$ -- Flags: -1 -3 Principal soln -1 Numeric results -3 -- Related: EXP,ISOL,LN,XROOT</p>
-------	----------------	--

1B374	xSQRT	<p>($x \rightarrow x'$) Square Root Analytic Func -- Returns the (+ve) square root of the argument. -- $z \rightarrow \sqrt{az}$ $x_u \rightarrow \sqrt{a}(x)_u$ 'sym' \rightarrow 'SQRT(sym)' -- Flags: -1 -3 --</p>
1FD61	xSIGMA+	<p>Related: SQ, ^, ISOL UserRPL: $x\sqrt{a}$ ($x \rightarrow$) ($x_1 \dots x_n \rightarrow$) ([] / [[]] \rightarrow) Sigma Plus Cmd -- Adds one or more data points to the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved ΣDAT --</p>
1FD8B	xSIGMA-	<p>Related: CLΣ, RCLΣ, STOΣ, Σ^- UserRPL: $x\Sigma^+$ ($\rightarrow x$) ($\rightarrow []$) Sigma Minus Cmd -- Returns a vector of m real numbers (or one number x if $m = 1$) corresponding to the coordinate values of the last data point entered by Σ^+ into the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved ΣDAT --</p>
1AABD	xPI	<p>Related: CLΣ, RCLΣ, STOΣ, Σ^+ UserRPL: $x\Sigma^-$ ($\rightarrow \pi$) PI Func -- Returns the symbolic constant 'pi' or its numerical representation, 3.14159265359. \rightarrow 'pi' \rightarrow 3.14159265359 -- Flags: -2 -3 -- Related: e, i, MAXR, MINR, \rightarrowQπ UserRPL: $x\pi$</p>

```

1ECFC      x<=?      ( x y → 1 )
              ( x y → 0 )
Less Than or Equal Func
--
Tests whether one object is less than or equal to
another object.
--
x          y          → 0/1
#n1       #n2        → 0/1
"str1"    "str2"     → 0/1
x         'sym'      → 'x<=sym'
'sym'    z          → 'sym<=z'
'sym1'   'sym2'     → 'sym1<=sym2'
x_u1     y_u2       → 0/1
x_u      'sym'      → 'x_unit<=sym'
'sym'    x_u        → 'sym<=x_unit'
--
Flags: -3
--
Related: <,>,≥,==,≠ UserRPL: x≤

1ED9B      x>=?      ( x y → 1 )
              ( x y → 0 )
Greater Than or Equal Func
--
x          y          → 0/1
#n1       #n2        → 0/1
"str1"    "str2"     → 0/1
x         'sym'      → 'x≥sym'
'sym'    z          → 'sym≥z'
'sym1'   'sym2'     → 'sym1≥sym2'
x_u1     y_u2       → 0/1
x_u      'sym'      → 'x_u≥sym'
'sym'    x_u        → 'sym≥x_u'
--
Flags: -3
--
Related: <,<=,>,==,≠ UserRPL: x≥

```


1EA9D	x#?	<p>(x y → 1) (x y → 0) Not Equal Func -- Tests if two objects are equal. obj1 obj2 → 0/1 (x,0) x → 0/1 x (x,0) → 0/1 z 'sym' → 'z≠sym' 'sym' z → 'sym≠z' 'sym1' 'sym2' → 'sym1≠sym2' -- Flags: -3 --</p>
234C1	xRPN->	<p>Related: SAME,TYPE,<,≤,>,≥, == UserRPL: x≠ (ob1 .. obn →) Create Local Variables Cmd -- Creates local variables. obj1 ... objn → -- Syntax: → name1 name2 ... nameN << prog >> → name1 name2 ... nameN 'Expr' --</p>
22FEB	xALG->	<p>Related: DEFINE,STO UserRPL: x→</p>
1BB02	xFACT	<p>Create local variable comand. <REF>xRPN-> UserRPL: x→ (x → x') Factorial (Gamma) Func -- Returns the factorial n! of a positive integer argument n, or the gamma function (x+1) of a non-integer argument x. n → n! x → (x+1) 'sym' → '(sym!)' -- Flags: -3 -20 -21 Numerical Results -3 Underflow exception -20 Overflow exception -21 -- Related: COMB,PERM UserRPL: x!</p>

```

1C060      x%      ( x y → xy/100 )
                Percent Func
                --
                Returns x (level 2) percent of y (level 1).
                x      y      → xy/100
                x      'sym' → '%(x,sym)'
                'sym' x      → '%(sym,x)'
                'sym1' 'sym2' → '%(sym1,sym2)'
                x      y_unit → (xy/100)_unit
                x_unit y      → (xy/100)_unit
                'sym' x_unit → '%(sym,x_unit)'
                x_unit 'sym' → '%(x_unit,sym)'
                --
                Flags:
                Numerical Results -3
                --
                Related: %CH,%T
1ADEE      x*      ( x y → x*y )
                Multiply Analytic Func
                --
                Returns the product of the args.
                z1      z2      → z1z2
                [[ mat ]] [ arr ] → [[ matarr ]]
                z      [ arr ] → [ z array ]
                [ arr ] z      → [ arr z ]
                z      'sym' → 'z * sym'
                'sym' z      → 'sym * z'
                'sym1' 'sym2' → 'sym1 * sym2'
                #n1     n2      → #n'
                n1     #n2     → #n'
                #n1     #n2     → #n'
                x_u     y_u     → xy_ux  unity
                x      y_u     → xy_u
                x_u     y      → xy_u
                'sym'   x_u     → 'sym * x_u'
                x_u     'sym'   → 'x_u * sym'
                --
                Flags: -3 -5 -6 -7 -8 -9 -10
                Numeric results -3
                bint wordsize -5 → -10
                --
                Related: +,-,/,=

```

1AB67 x+

(x y → x+y)
Add Analytic Func

--

Returns the sum of the arguments. Addition. If one arg is list, insert element in list or concatenate lists.

<REF>xADD

```

z1      z2      → z1+z2
[ arr ]1 [ arr ]2 → [ arr ]1+2
z      'sym'    → 'z+(sym)'
```

'symb' z → 'sym+z'

'sym1' 'sym2' → 'sym1 + sym2'

{ lst1 } { lst2 } → { lst1 lst2 }

obj { o... } → { obj o... }

{ o... } o → { o... obj }

"str1" "str2" → "str1str2"

obj "str" → "obj str"

"str" obj → "str obj"

#n1 n2 → #n'

n1 #n2 → #n'

#n1 #n2 → #n'

x1_u1 y_u2 → (x2+y)_u2

'sym' x_u → 'sym+x_u'

x_u 'sym' → 'x_u+sym'

grob1 grob2 → grob'

--

Flags: -3 -4 -5 -6 -7 -8 -9 -10

Numeric results -3

Bint wordsize -5 → -10

--

Related: -,*,/,=

```

1AD09      x-      ( x y → x-y )
              Subtract Analytic Func
              --
              Returns the difference of the arguments: the object
              in level 1 is subtracted from the object in level 2.
              z1      z2      → z1-z2
              [ arr ]1 [ arr ]2 → [ arr ]1_2
              z      'sym'   → 'z-sym'
              'sym'  z      → 'sym-z'
              'sym1' 'sym2'  → 'sym1 - sym2'
              #n1    n2      → #n'
              n1     #n2     → #n'
              #n1    #n2     → #n'
              x1_u1  y_u2    → (x2-y)_u2
              'sym'  x_u     → 'sym-x_u'
              x_u    'sym'   → 'x_u-sym'
              --
              Flags: -3
              Numeric results -3
              --
              Related: +,*,/,=
1AF05      x/      ( x y → x/y )
              Divide Analytic Func
              --
              Returns the quotient of the arguments: the level 2
              object divided by the level 1 object. (Abbrev. _u =
              _unit)
              z1      z2      → z1 / z2
              [ arr ] [[ mat ]] → [[mat^-1arr]]
              [ arr ] z      → [ arr / z ]
              z      'sym'   → 'z / sym'
              'sym'  z      → 'sym / z'
              'sym1' 'sym2'  → 'sym1 / sym2'
              #n1    n2      → #n'
              n1     #n2     → #n'
              #n1    #n2     → #n'
              x_u1   y_u2    → (x/y)_u1/u2
              x      y_u     → (x/y)_1/u
              x_u    y      → (x/y)_u
              'sym'  x_u     → 'sym/x_u'
              x_u    'sym'   → 'x_u/sym'
              --
              Related: +,-,*,=,RATIO

```

```

1EBBE      x<      ( x y → 1 )
              ( x y → 0 )
              Less Than Func
              --
              Tests whether one object is less than another object.
              x      y      → 0/1
              #n1    #n2    → 0/1
              "str1" "str2" → 0/1
              x      'sym'  → 'x<sym'
              'sym'  x      → 'sym<z'
              'sym1' 'sym2' → 'sym1<sym2'
              x_u1   y_u2   → 0/1
              x_u    'sym'  → 'x_u<sym'
              'sym'  x_u    → 'sym<x_u'
              --
              Flags: -3
              Numeric results -3
1A8D8      x=      ( x y → x=y )
              Makes equation out of two expressions. Equals An-
              alytic Func
              --
              Returns an equation formed from the two arguments.
              z1     z2     → 'z1=z2'
              z      'sym'  → 'z=sym'
              'sym'  z      → 'sym=z'
              'sym1' 'sym2' → 'sym1=sym2'
              y      x_u    → 'y=x_u'
              y_u    x      → 'y_u=x'
              y_u    x_u    → 'y_u=x_u'
              'sym'  x_u    → 'sym=x_u'
              x_u    'sym'  → 'x_u=sym'
              --
              Flags: -3
              Numeric results -3
              --
              Related: DEFINE,EVAL,-

```

```

1E972      x==      ( x y → 1 )
              ( x y → 0 )
              Logical Equality Func
              --
              Tests if two objects are equal.
              obj1  obj2  → 0/1
              (x,0) x     → 0/1
              x     (x,0) → 0/1
              z     'sym' → 'z==sym'
              'sym' z     → 'sym==z'
              'sym1' 'sym2' → 'sym1==sym2'
              --
              Flags: -3
              Numeric results -3
              --
              Related: SAME,TYPE,<,<=,>,>=,≠

1EC5D      x>      ( x y → 1 )
              ( x y → 0 )
              Greater Than Func
              --
              Tests whether one object is greater than another ob-
              ject.
              x     y     → 0/1
              #n1  #n2   → 0/1
              "str1" "str2" → 0/1
              x     'sym' → 'x>sym'
              'sym' z     → 'sym>z'
              'sym1' 'sym2' → 'sym1>sym2'
              x_u1  y_u2  → 0/1
              x_u   'sym' → 'x_u>sym'
              'sym' x_u   → 'sym>x_u'
              --
              Flags: -3
              Numeric results -3
              --
              Related: <,<=,>,>=,==,≠ ;

```

7 ML Entry Points

7.1 General Purpose

0679B	SAVPTR	D0 to RPLTOP D1 to DSKTOP B to RETTOP D to FREETOP Clear carry
067D2	GETPTR	<see>SAVPTR in reverse Clears Carry.
05143	GETPTRLLOOP	<see>GETPTR , Loop to RPL
6384E	D0=DSKTOP	Get new D0 from DSKTOP, uses A
6385D	D1=DSKTOP	Get new D1 from DSKTOP, uses C
010E5	AllowIntr	Allow interrupts.
01115	DisableIntr	Disable interrupts.
0115A	AINRTN	A=IN see also <see>CINRTN For hardware reasons (bug)
01160	CINRTN	A=IN must be at even addr C=IN see also <see>AINRTN For hardware reasons (bug) C=IN must be at even addr

7.2 Errors

7.2.1 Generating Errors

04FBB	DOMEMERR	Insufficient Memory error
18CA7	DOSIZEERR	Bad Argument Value error
05023	Errjmp	Error exit A.A = error number
10F80	ErrjmpC	A=C.A <see>Errjmp
10F40	GPErrjmpC	A=C.A <see>GETPTR <see>Errjmp
065AA	GPMEMERR	<see>GETPTR <see>DOMEMERR

7.2.2 Error Number Constants

00202	argtypeerr	"Bad Argument Type"
00203	argvalerr	"Bad Argument Value"

00B02	constuniterr	"Inconsistent Units"
00305	infreserr	"Infinite Result"
00A03	intrptderr	"Interrupted"
00C14	lowbaterr	"Low Battery"
00302	negunferr	"Negative Underflow"
00303	ofloerr	"Overflow"
0000A	portnotaverr	"Port Not Available"
00301	posunferr	"Positive Underflow"
00C13	prtparerr	"Invalid PRTPAR"
00C02	timeouterr	"Timeout"
00C06	xferfailerr	"Transfer Failed"

7.3 Hexadecimal Math

0D62F	DCHXW	Converts BCD in C.W to hex in A.W B.W C.W. See <see>HXDCW Uses P CRY
06A8E	DIV5	C.A = C.A/5 Uses A.10 C.10 D.10 P
0DB91	HXDCW	Converts hex in A.W to BCD in A.W B.W C.W. See <see>DCHXW Uses P CRY Note that HXDCW wants the input in A but DCHXW wants it in C
03F24	IntDiv	A.A/C.A -> A.A=remainder, C.A=quotient, uses D.A P SB
53EE4	MPY	Multiply A.W and C.W (-> A.W=C.W) Uses D.W, SB. Returns carry clear
03991	MUL#	B.A = A.A*C.A

7.4 Long Reals

7.4.1 Storage Handling

2BE61	STAB0	A.W -> R0 B.W -> R1
2BE6F	STAB2	A.W -> R2 B.W -> R3
2BE7D	STCD0	C.W -> R0 B.W -> R1
2BE8B	STCD2	C.W -> R2 B.W -> R3

2BEB5	RCAB0	R0 -> A.W R1 -> B.W
2BEC0	RCAB2	R2 -> A.W R3 -> B.W
2BECB	RCCD0	R0 -> C.W R1 -> D.W
2BED6	RCCD2	R2 -> C.W R3 -> D.W
2BE99	EXAB0	A.W <-> R0 B.W <-> R1
2BEA7	EXAB2	A.W <-> R2 B.W <-> R3
2BE53	XYEX	A:B <-> C:D

7.4.2 Calculating

2B977	DIVF	x=x/y
2B91E	MULTF	x=x*y
2B7B0	RADD1	x=x+1 see <see>RADDf
2B7CA	RADDf	x=x+y
2B7A7	RSUB1	x=x-1 see <see>RADDf

7.4.3 Conversion

29E46	PACK	(x -> A) <see>PACKSB without rounding
29E21	PACKSB	(x -> A) Converts %% to %. If SB is clear uses roundup, if set uses lowest nibble in % field to determine rounding direction. Obeyes and sets flow flags/indicators
2BC4A	SPLITA	(A -> x) Convert % to %%
2BCA0	SPLTAC	(A,C -> x, y) Convert 2 reals to long reals

7.5 Memory Handling

7.5.1 General Memory Handling Routines

069F7	ADJMEM	D= @FREETOP=<see>ROOM / 5 Uses A.10 B.10 C.10 D.10 <see>DIV5
-------	--------	---

0554C	DOGARBAGE	If ST=1 10 then <see>GPMEMERR else <see>GARBAGECOL and <see>GETPTR
0613E	GARBAGECOL	Garbage collection does not use R1..R4
06806	ROOM	-> C.A = @DSKTOP-@RETTOP Uses A.A D0
03019	SKIPOB	Skip object in D0, clears ST1, clears carry, P=0 --> D0 = addr past object Uses: A.A C.A P ST1 RSTK2

7.5.2 Moving and Swapping Memory Areas

0670C	MOVEDOWN	Copy downwards C.A nibbles from D0 to D1, D0 and D1 will point to the next locations Used: A.W C.A P Use this to move upwards
06992	MOVERSD	Delete a block below RSK A.A=end C.A=nibbles Adjusts all refs, then <see>ADJMEM
06A53	MOVERSU	Uses A.W B.A C.W D.10 D0 D1 P Open a block below RSK A.A=start C.A=nibbles Adjusts all refs, then <see>ADJMEM
06A1D	MOVEDSD	Uses A.W B.A C.10 D.10 D0 D1 P Open a block above stack A.A=end C.A=nibbles Adjusts all refs, then <see>ADJMEM
069C5	MOVEDSU	Uses A.W B.A C.10 D.10 D0 D1 P Delete a block above stack A.A=start C.A=nibbles Adjusts all refs, then <see>ADJMEM
066B9	MOVEUP	Uses A.W B.A C.10 D.10 D0 D1 P Copy upwards C.A nibbles from D0 to D1 D0 D1 will point to start of area Used: A.W C.A P Use this to move downwards

7.5.3 Allocating Memory in TEMPOB

06AD8	CREATETEMP	Allocates C.A nibbles carry if not enough memory -> D0=bottom, D1=top of area -> B.A = C.A = @D1 = offset to previous tempob = #nibbles+6
039BE	GETTEMP	<see>CREATETEMP with <see>GARBAGECOL if necessary <see>GPMEMERR if not enough memory
05B79	MAKE\$	Creates character string in tempob area Does SETHEX, C=C+C.A and then <see>MAKE\$N
05B7D	MAKE\$N	Creates character string in tempob area If not enough mem even after GC then memerr C.A = nibbles -> A=nibbles+5, B=nibbles+16 C=D1=addr of stack D0 = addr of body of \$ R0 = addr of \$ Not used: R1-R4

7.5.4 Resizing TEMPOB Areas

16683	(Clean\$)	Shrink strobj in top of TEMPOB R1=addr of length field A.A=new end address Uses A.W B.A C.W D.A D0 D1
16677	(Clean\$R0)	R1=R0+5 <see>Clean\$
16671	Shrink\$	Shrinks a strobj R0.A=->\$ D0=end of \$ Uses A.W B.A C.W D.10 D0 D1

7.5.5 CRC Routines

05981	DoCRC	Calculates the CRC of A.A nibs at D0. Returns CRC in A.A Uses C.W P Turns interrupts off and on
0597E	DoCRCc	D0=C <see>DoCRC
0A00E	CKLBCRC	Check CRC of library at D0 CC: Ok CS: CRC is wrong Uses A.A C.W DO P Disables and re-enables interrupts

00108	(POWERSTATUS)	Low power registers
00109	(POWERCTRL)	Low power detection
00137	TIMER1	1 nibble timer decremented 16 times/s
00138	TIMER2	8 nibble timer decremented 8192 times/s
0012E	(TIMER1CTRL)	TIMER1 control [SRQ WKE INT XTRA]
0012F	(TIMER2CTRL)	TIMER2 control [SRQ WKE INT TRUN]

7.7 Display

11D8F	\$5x7	(D.A B.A C.A D0 D1 -->) Displays string body at D1 in grob at D0 C.A = chars B.A = xlocation D.A = row length in nibbles -> D1 = addr after \$ D0 = location of next char D.A = row length
01C31	D0->Row1	(--> D0) Gets addr of current display
01C58	D0->Sft1	(--> D0) Gets address of menu grob
01B8F	DispOn	Turns display on <see>Dispoff
01BBD	DispOff	Turns display off <see>Dispon
116B5	(grob!)	R0,R1 = (row,col), D0 = grob1, D1 = grob2 --> Stores grob1 into grob2
115B3	makegrob	R0.A = x, R1.A = y --> D0 = body Makes a grob of size x,y Prolog is in D0-20
1165A	w->W	Calculates GROB width A.A=width in pixels -> A.A=width in nibbles Basically the same as $8 / \text{CEIL } 2 *$ since the width must be an even number of nibbles

7.8 Popping and Pushing

7.8.1 Pointers

03249	DropLoop	Pop stack, Loop
60F83	4DropLoop	Pop 4, Loop
03672	GPOverWrALp	<see>GETPTR , OverWr A, Loop
0366F	GPOverWrROLp	<see>GETPTR , OverWr R0, Loop
54266	GPPushA	<see>GETPTR , Push A, Clear Carry
3251C	PopASavptr	Pop to A.A, <see>SAVPTR
3251F	PopSavptr	Pop <see>SAVPTR
03A86	PUSHA	Push A, Loop

7.8.2 TRUE and FALSE

26FAE	GETPTRFALSE	<see>GETPTR , Do FALSE
25CE1	GETPTRTRUE	<see>GETPTR , Do TRUE
62096	GPOverWrFLp	<see>GETPTR , OverWr FALSE, Loop
62076	GPOverWrTLp	<see>GETPTR , OverWr TRUE, Loop
62073	GPOverWrT/FL	<see>GETPTR , OverWr TRUE/FALSE, Loop
620D2	GPPushFLoop	<see>GETPTR , Push FALSE, Loop
267D5	GPPushFTLp	<see>GETPTR , Loop to FalseTrue
620B9	GPPushTLoop	<see>GETPTR , Push TRUE, Loop
620B6	GPPushT/FLp	<see>GETPTR , Push TRUE/FALSE, Loop
620A0	OverWrFLoop	OverWr FALSE, Loop
62080	OverWrTLoop	OverWr TRUE, Loop
6209D	OverWrT/FLp	OverWr TRUE/FALSE, Loop
61A02	popflag	Pop to A.A, if TRUE then set carry
620DC	PushFLoop	Push FALSE, Loop
620C0	PushF/TLoop	Push FALSE (CRY)/TRUE, Loop
620C3	PushTLoop	Push TRUE, Loop
620D9	PushT/FLoop	Push TRUE (CRY)/FALSE, Loop aka: PushT/F
267DC	(PushFTLp)	Loop to False/True

7.8.3 System Binary Integers (BINT)

06641	POP#	Pop # to A.A
03F5D	POP2#	(#1 #2 -->) Pop #1 to A.A and #2 to C.A
06537	PUSH#	<see>GETPTR , Push R0 as #

03DC7	Push#Loop	<see>SAVPTR , R0=A, <see>PUSH# , Loop
06529	PUSH2#	<see>GETPTR , Push R0 & R1 as #
0357F	PUSH#LOOP	<see>GETPTR , Push R0 as #, Loop
0357C	PUSH#AL00P	<see>GETPTR , Push A as #, Loop
03F14	Push2#Loop	<see>GETPTR , Push R0 & R1 as #, Loop
627EB	Push2#aLoop	<see>GETPTR , Push R0 & A as #, Loop
036F7	Push#TLoop	<see>GETPTR , Push R0 as #, Do TRUE
2E31F	Push#FLoop	<see>GETPTR , Push R0 as #, Do FALSE

7.8.4 HXS and ZINTs

53F8D	(POPHXS)	Pop hxs to A, <see>SAVPTR , Clear Carry
53F77	(POP2HXS)	Pop hxs to C and hxs to A, <see>SAVPTR , Clear Carry P has current wordsize.
5422C	PUSHhxs	Push A.WP as hxs
0596D	PUSHhxsLoop	Push A.WP as hxs, Loop

7.8.5 Real and Complex Numbers

29FD0	POP1%SPLITA	(%pop -> x) Pop %, convert to %, <see>SAVPTR
29FDA	POP1%	(%pop -> A) Pop %, <see>SAVPTR
2A002	POP2%	(%pop1 %pop2 -> A,C) Pop 2 reals, <see>SAVPTR
2A188	PUSH%	(A -> %push) Push A as %, <see>GETPTR
2A23D	PUSH%LOOP	(A -> %push) Push A as %, <see>GETPTRLOOP
52AB7	(POPC%)	(C%pop -> A:C) Pop C% (<see>SETDEC)
52B57	(POPC%%)	(C%%pop -> A:B C:D) Pop C%% (<see>SETDEC)
52ADB	(PUSHC%)	(A:C -> C%push) Push C%
52B95	(PUSHC%%)	(A:B:C:D -> C%%push) Push C%%

7.9 Keyboard Handling

04988	(Attn?)	Sets carry when ATTNFLG <> 0.
0CA60	ATTNchk	ATTN exit check with restoreiram
009A5	Debounce	Scans keyboard until no more instabilities detected returns a map of the pressed keys in A.W 48G[X+] Keymap nibbles: (Nibble: [Bit1 Bit2 Bit3 Bit4]) 0: [ON + SPC .] 1: [0 ' - 3] 2: [2 1 A RS] 3: [* 6 5 4] 4: [MTH LS / 9] 5: [8 7 SIN alpha] 6: [BackSp DEL EEX +/-] 7: [ENTER 1/x y^x SQRT] 8: [TAN COS right down] 9: [left EVAL STO NXT] A: [up VAR CST PRG] B: [F E D C] C: [B none none none] 49G Keymap nibbles: 0: [ON RS LS alpha] 1: 2: [right down left up] 3: 4: [A B C D] 5: [E F none APPS] 6: 7: [EEX y^x HIST MODE] 8: [0 1 4 7] 9: [+/- SQRT CAT TOOL] A: [. 2 5 8] B: [1/x SIN EQW VAR] C: [SPC 3 6 9] D: [X COS SYMB STO] E: [ENTER + - *] F: [/ TAN BackSp NXT]
04999	KeyInBuff?	Carry if true
00C74	OnKeyDown?	Carry if true
00C80	OnKeyStable?	Carry if true
00D57	Flush	Flushes key buffer.
00D8E	FlushAttn	Flushes attn counter.

04840	POPKEY	(-> C.A) Sets carry if buffer is empty.Else returns key in C.B (and in @KEYSTORE) Uses: A.S C.S C.A D1 (sets P=0)
007B5	SrvcKbdAB	(A.W ->) Sets KEYSTATE and KEYBUFFER

7.10 Various ML Entries

0D5E5	ASRW5	ASR.W 5 times
0D5F6	ASLW5	ASL.W 5 times
2BEE1	CCSB1	Uses D.S to set SB, clears carry
018E2	clkspd	Measure CPU clock speed Interrupts off on entry and exit -> A.A=spd/16 B.A=loops/16s Uses C.A D0 P CRY
0D607	CSRW5	CSR.W 5 times
0D618	CSLW5	CSL.W 5 times
04292	DeepSleep	Puts calc into "deep sleep" Low power mode, display off Wakeup on ON key or interrupt
5F090	(doskip)	Exit to rpl SKIP
5F09D	(docola)	Exit to rpl COLA
2FFB4	GetStrLenStk	Pop \$ -> C.A = length, D1 = body
2FFB7	GetStrLenC	D1 = C, <see>GetStrLen
2FFBA	GetStrLen	D1=\$ -> C.A = length, D1 = body
54021	(getwordsize)	Fetches current word size to A, Clear Carry
017A6	makebeep	C = msec, D = Hz Checks BEEP flag.
04929	liteslp	Puts calc into "lite sleep" Low power mode with display on Wakeup on any key or interrupt

7.11 Object Types

029E8	DOARRAY	Array prologue 5 size 5 prologue of objects 5 # of dimensions 5n dimensions .. objects (content only)
-------	---------	--

02B62	DOBAK	Backup prologue 5 size 2 # of chars in name .. name .. object 5 DOBINT 5 CRC Apparently unused on the 49
02911	DOBINT	BINT prologue 5 number (hex)
029BF	DOCHAR	Character prologue 2 character
02977	DOCMP	Complex number prologue 3 real exponent 12 real mantissa 1 real sign 3 complex exponent 12 complex mantissa 1 complex sign
02DCC	DOCODE	Code prologue 5 length
02D9D	DOCOL	.. machine code Secondary prologue .. objects 5 SEMI
02A2C	DOCSTR	String prologue 5 length .. characters
0299D	DOECMP	Long complex prologue 5 real exponent 15 real mantissa 1 real sign 5 complex exponent 15 complex mantissa 1 complex sign
02955	DOEREL	Long real prologue 5 exponent 15 mantissa 1 sign
02ADA	DOEXT	Unit object prologue .. object (usually a real) .. unit
02B1E	DOGROB	5 SEMI GROB prologue 5 size 5 height 5 width

02A4E	DOHSTR	HXS prologue 5 length .. hex digits, reverse order aka: DOHXS
02E48	DOIDNT	Global name (ID) prologue 2 # of characters .. characters
02E6D	DOLAM	Local name (LAM) prologue see <see>DOIDNT
02A0A	DOLNKARRY	Linked array prologue Not used by the system.
02B40	DOLIB	Library prologue 5 size 2 # of characters .. name 2 # of characters (unless 0) 3 library ID 5 hash table offset 5 message table offset 5 link table offset 5 config object offset .. contents 4 CRC ; XLIBs: 1 or 3: kind 3 library ID 3 command ID .. object -- <REF>TEXT:Libraries
02A74	DOLIST	List prologue see <see>DOCOL
02933	DOREAL	Real number prologue 3 exponent 12 mantissa 1 sign
02E92	DOROMP	XLIB prologue 3 library ID 3 command #

02A96	DORRP	Directory prologue Home directory: 3 # of attached libs n* [3 library ID 5 address of hash table 5 address of message table] 5 offset of last object * [5 offset to previous object 00000 for the first one 2 # of characters .. name of object 2 # of characters .. object] ; Subdirectories: 3 # of attached library 7FF if none 5 offset of last object .. same as above
02AB8	DOSYMB	Symbolic prologue .. objects
02AFC	DOTAG	5 SEMI Tagged object prologue 2 # of chars in tag .. tag .. object
02B88	DOEXT0	
02BAA	DOEXT1	
		aka: DOACPTR
02BCC	DOEXT2	
02BEE	DOEXT3	
02C10	DOEXT4	

8 RAM entries

Note that pointers (->...) are always 5 nibbles wide.

8.1 RPL pointers

The contents of the following four locations are only valid after SAVPTR.

807ED	AVMEM	Free mem / 5 (5)
806F8	DSKTOP	->Data stack
806F3	RSKTOP	->Return stack
8072F	INTRPPTR	->RPL runstream aka: OBUPSTART

8.2 Memory management pointers

806E9	TEMPOB	->Beginning of TempOb area
806EE	TEMPTOP	->End of TempOb area
80711	USEROB	->UserOb Area (HOME)

8.3 Screen related

806D5	ADISP	->Stack grob
806E4	GDISP	->Blackboard grob
806DA	VDISP	->Display grob aka: VDISP1, SYSUPSTART
806D0	VDISP2	->Menu grob
806DF	VDISP3	->Not displayed grob <see>VDISP

8.4 Annunciators

80841	ANNUNCIATORS	Annunciator flags (2)
00004	Ann_Susp.b	
00010	Ann_Shift.b	
00001	Ann_RAD.b	
00002	Ann_IO.b	
00080	Ann_Busy.b	
00040	Ann_Alpha.b	

00008 Ann_Alert.b
 00020 Ann_ALT.b

8.5 Save areas

805DB	INTRAM	Save area for the interrupt sys (16)
806C0	R1[A]save	Used by PrintLCD inside the interrupt system (5)
806BA	R2[A]save	<see>R1[A]save (5)
806BF	R2[S]save	<see>R1[A]save (1)
80831	SAVECLK	Save of CLKON state (1)
808D8	SAVECROSS	cursor moves in plotting (10)
805F5	SAVE_A	<see>INTRAM (16)
80608	SAVE_B	<see>INTRAM (16)
805F0	SAVE_C[A]	<see>INTRAM (5)
806C5	SAVE_BO	Save BitOffset (1)
80618	SAVE_D	<see>INTRAM (16)
8063D	SAVE_DO	<see>INTRAM (5)
806C6	SAVE_LC	Save LineCount (2)
806C8	SAVE_LN	Save LineNibs (3)
805EB	SAVE_MODES	<see>INTRAM (5)
806CB	SAVE_OFFSET	Save Window Offset (5)
80638	SAVE_PC	<see>INTRAM (5)
80628	SAVE_RO	<see>INTRAM (16)
80605	SAVE_ST	<see>INTRAM (3)
8069C	Stk0save	RSTK0 used by PrintLCD inside the interrupt sys (5)
806A1	Stk1save	RSTK1 <see>Stk0save (5)
806A6	Stk2save	RSTK2 <see>Stk0save (5)
806AB	Stk3save	RSTK3 <see>Stk0save (5)
806B0	Stk4save	RSTK4 <see>Stk0save (5)
806B5	Stk5save	RSTK5 <see>Stk0save (5)

8.6 System and User Flags

80843	SystemFlags	128 System flags (16)
80853	UserFlags	128 User Flags (16)

8.7 Internal System Flags

Unless otherwise indicated, the description of each MASK shows what this bit means if it's set.

80801	SysNib1	ISysFlags 1
00101	NoRolDA2MASK	DA2 can't be rolled up to become valid <see>SysNib1
00201	AbbrStkMASK	Display obj types only <see>SysNib1
00401	DA2bIsEdMASK	DA2b shows the edit line <see>SysNib1
00801	IgnorAlmMASK	Ignore <see>ALARMSDUE in <see>GETKEY <see>SysNib1
80802	SysNib2	ISysFlags 2
00102	ReqClkOnMASK	Flag for System Request of CLKON state <see>SysNib2
00202	ServModeMASK	Server mode on <see>SysNib2
00402	TrackMASK	New context needs to be compared with old <see>SysNib2
00802	BadMenuMASK	Menu system corrupt <see>SysNib2
80803	SysNib3	ISysFlags 3
00103	UNDOMASK	Automatic stack save <see>SysNib3
00203	INSERTMASK	Insert/replace mode <see>SysNib3
00403	ALGMASK	Algebraic entry mode <see>SysNib3
00803	PRINTINGMASK	<see>SysNib3
80804	SysNib4	ISysFlags 4
00104	DA2aTempMASK	DA2a temporarily valid <see>SysNib4
00204	DA2bTempMASK	DA2b temporarily valid <see>SysNib4
00404	DA3TempMASK	DA3 temporarily valid <see>SysNib4
00804	RebuildMASK	Menu requires TOUCHTAB rebuild each time it is redisplayed <see>SysNib4
80805	SysNib5	ISysFlags 5
00105	COMMANDMASK	CMD history enabled <see>SysNib5
00205	BLINKMASK	Active Timer1 Int's <see>SysNib5
00405	LOWERMASK	Lowercase keys <see>SysNib5
00805	STKDCMASK	Decompilation for stack display (not editing) <see>SysNib5
80806	SysNib6	ISysFlags 6
00106	Do1UserMASK	One-key user mode <see>SysNib6

00206	ASuspOKMASK	Suspending current environment is allowed <see>SysNib6
00406	BadPOLUIMASK	POL UI possibly corrupt <see>SysNib6
00806	DA1TempMASK	DA1 temporarily valid <see>SysNib6
80807	SysNib7	ISysFlags 7
00107	DA1ValidMASK	DA1 known to be valid <see>SysNib7
00207	DA2aValdMASK	DA2a known to be valid <see>SysNib7
00407	DA2bValdMASK	DA2b known to be valid <see>SysNib7
00807	DA3ValidMASK	DA3 known to be valid <see>SysNib7
80808	SysNib8	ISysFlags 8
00108	DA1NoChMASK	DA1 not changed <see>SysNib8
00208	DA2aNoChMASK	DA2a not changed <see>SysNib8
00408	DA2bNoChMASK	DA2b not changed <see>SysNib8
00808	DA3NoChMASK	DA3 not changed <see>SysNib8
80809	SysNib9	ISysFlags 9
00109	DA1BadMASK	DA1 invalid <see>SysNib9
00209	DA2aBadMASK	DA2a invalid <see>SysNib9
00409	DA2bBadMASK	DA2b invalid <see>SysNib9
00809	DA3BadMASK	DA3 invalid <see>SysNib9
8080A	SysNib10	ISysFlags 10 aka: EDITFLAG, EDITLFLAG
0010A	EDITLMASK	Edit line exists <see>SysNib10
0020A	NAppKeyMASK	Non-app keys allowed in POL <see>SysNib10
0040A	NUsrKeyMASK	Non-user keys allowed in USR mode <see>SysNib10
0080A	AppModeMASK	POL application running <see>SysNib10
8080B	SysNib11	ISysFlags 11 aka: ParenModFLAG
0010B	ParenModMASK	Implicit parenthesized "/", "^", and "SQRT" in EQW <see>SysNib11
0020B	1PDCMASK	Partial DeCompile info will not be saved <see>SysNib11
0040B	NewEditLMASK	New one-line edit line has been created <see>SysNib11
0080B	DoStdKeyMASK	Do only standard keys <see>SysNib11
8080C	SysNib12	ISysFlags 12
0010C	DispTimeMASK	Status bar clock may be displayed <see>SysNib12
0020C	NOP2MASK12	unused <see>SysNib12
0040C	NOP4MASK12	unused <see>SysNib12
0080C	NOP8MASK12	unused <see>SysNib12

8080D	SysNib13	ISysFlags 13
0010D	NOP1MASK13	unused <see>SysNib13
0020D	NOP2MASK13	unused <see>SysNib13
0040D	NOP4MASK13	unused <see>SysNib13
0080D	NOP8MASK13	unused <see>SysNib13
8080E	SizeMLDisp	
		aka: SysNib14
0010E	NOP1MASK14	unused <see>SysNib14
0020E	NOP2MASK14	unused <see>SysNib14
0040E	NOP4MASK14	unused <see>SysNib14
0080E	NOP8MASK14	unused <see>SysNib14
8080F	SysNib15	ISysFlags 15
001CE	BadTOLUIMASK	TOL UI potentially corrupt <see>SysNib15
		aka: NOP1MASK15
002CE	NoAlgProcess	EVAL-> will not create a list nor return NOVAL <see>SysNib15
		aka: NOP2MASK15
004CE	InSimplyExpr	<see>SysNib15
		aka: NOP4MASK15
008CE	DoCreateMenu	<see>SysNib15
		aka: NOP8MASK15
80810	SysNib16	ISysFlags 16 (unused)
00110	NOP1MASK16	<see>SysNib16
00210	NOP2MASK16	<see>SysNib16
00410	NOP4MASK16	<see>SysNib16
00810	NOP8MASK16	<see>SysNib16
80811	SysNib17	ISysFlags 17 (unused)
00111	NOP1MASK17	<see>SysNib17
00211	NOP2MASK17	<see>SysNib17
00411	NOP4MASK17	<see>SysNib17
00811	NOP8MASK17	<see>SysNib17
80812	SysNib18	ISysFlags 18 (unused)
00112	NOP1MASK18	<see>SysNib18
00212	NOP2MASK18	<see>SysNib18
00412	NOP4MASK18	<see>SysNib18
00812	NOP8MASK18	<see>SysNib18
80813	SysNib19	ISysFlags 19 (unused)
00113	NOP1MASK19	<see>SysNib19
00213	NOP2MASK19	<see>SysNib19
00413	NOP4MASK19	<see>SysNib19

00813	NOP8MASK19	<see>SysNib19
80814	SysNib20	ISysFlags 20 (unused)
00114	NOP1MASK20	<see>SysNib20
00214	NOP2MASK20	<see>SysNib20
00414	NOP4MASK20	<see>SysNib20
00814	NOP8MASK20	<see>SysNib20

8.8 Warmstart log

80010	FAILSTK1	Warmstart log 1st (newest) entry (18) Each entry consists of a one-nibble cause (as displayed by WSLOG), a 13-nibble time stamp and a 4-nibble CRC of the previous 14 nibbles.
80022	FAILSTK2	<see>FAILSTK1 2nd entry (18)
80034	FAILSTK3	<see>FAILSTK1 3rd entry (18)
80046	FAILSTK4	<see>FAILSTK1 4th entry (18)

8.9 Command line management

8086A	CR_COUNT	# of newlines in editline (5)
80882	CURSOR	Cursor editline position (5) aka: CURSOREPOSN
8088F	CURSORCHR	Char under Cursor (2)
80891	CURSORGROB	Cursor Grob Data (40)
8088C	CURSOROFFSET	Cursor position from left of screen (2) aka: CURSORPOSN
80887	CURSORPART	Cursor display row (5) aka: CURSORROW
8088E	CURSORSTATE	Show cursor/char underneath (1)
808B9	CURSORX	Pxl X-Coord of Cursor (5)
808BE	CURSORY	Pxl Y-Coord of Cursor (5)
806FD	EDITLINE	->Command line

8.10 POL variables

80815	AppCount	# of nested POLs (2)
80784	AppDisplay	->App display object

80793	AppError	->App error handler
8078E	AppExitCond	->App exit condition
80789	AppKeys	->App key assignments

8.11 UART buffering

80519	uart_buf_end	# of bytes in the UART buffer (2)
8051C	uart_buf_st	UART buffer offset (2)
80319	uart_buffer	UART buffer area (512)
8051B	uart_error	UART error flag (1)
8051E	uart_handshk	UART handshake (1)
8051F	uart_modes	UART mode (1)
80520	uart_parity	(1)
80521	uart_timeout	(2)

8.12 ROM Part Tables

809A3	ROMPTAB	Library table (3+n*16) Header: 3 number of libraries For each library: 3 library ID 5 address 5 switch routine (0 if none) 3 000 aka: RESRAMEND, FlashROMPTAB
-------	---------	--

8.13 Constants

The entries in this section do not denote actual memory addresses, but constants related to them.

00008	IRAMHOMEmsn	MSN of the IRAM base address
00011	LOCUPSIZE	Number of variables between <see>SYSUPSTART and <see>OBUPSTART
00025	OBUPSIZE	Number of variables between <see>OBUPSTART and <see>OBUPEND
00001	mEditLExists	
00036	SYSUPSIZE	aka: ParenModmask <see>OBUPSIZE + <see>LOCUPSIZE

8.14 Other/Uncategorized

80912	ACCUM	(1)
8072A	ALARMS	->System Alarm List (5)

80832	ALARMSDUE	Flags Alarm Due (1)
80836	ALARM#	
807F7	ATTNFLG	Counts ON presses (5)
800E6	AccessInit	Saved value of INITEN & sALLOWINTR (2)
80000	CMOS	Quick RAM corrupt check (5) aka: HARDROMEND, RAMSTART
80922	COLCOUNT	Dot Cols on line (2)
80914	COLWIDTH	(2)
80524	CONFRAM	RAM configuration (7) Port1: 1 Status [r w s 0] 1 Size/Address Code Port2: 1 Status [r w s 0] 1 Size/Address Code where r=readable, w=writable, s=system RAM 2 #banks 1 ID
8052B	CONFTAB	RAM configuration with CRC (11) 4 nibbles for CRC 7 nibbles as in CONFRAM
8071B	CONTEXT	->Current dir
800EB	COVERsave	Save area for G/DoCovered (10)
800E8	COVERstate	Iram state before uncovering (3)
80076	TIMEOUTCLK	ScratchPad (4)
80655	CSPEED	CPU speed (16hz units) (5)
808C3	CURRENTMENU	Menu ID of current menu (2)
8091B	ClkOnNib	Clock display on/off (1)
80798	CtlAlarm	->Control alarm data
8081D	DEPTHSAVE	Saved user stack depth (5)
8065B	DISABLE_KBD	Keyboard handshake (1) aka: HANDSHK
8068D	DISP1CTLg	Ghost for DISP1CTL (5)
80695	DISP2CTLg	Ghost for DISP2CTL (5)
80707	DOLPENV	->DO LOOP environments
80834	DOUSEALARM	Flags Deactivate Curr Alarm (1)
8064A	DREND	Display Refresh Hi Bound (5)
80645	DRSTART	Display Refresh Lo Bound (5)
808EE	DcompWidth	String Decomp Width (2)
8091E	DelayCt	REDEYE Print time/line (2)
80863	ELEMENT	decompile obj depth counter (2)
80916	ENTRWISE	(1)
807F2	ERROR	(5)
8077F	EXITMSG	->msg set by user in EXIT word
8094C	EqPtr	Points to Curr Eqn in EqList (5)
80865	FIRSTCHAR	offset to 1st visible (5)

807FC	FIRSTPROC	->StartupProc Secondary (5)
808F2	FONTCOUNT	counter (3)
808F0	FONTHEIGHT	font-height selector (1)
808F1	FONTWIDTH	font-width selector (1)
80085	FailTime	SelfTest Fail Time (Ticks) (13)
8092A	FifoByteCt	Sum of FIFO Line Counts (2)
808CE	GARBSCRATCH1	Saves 1 RSTK level in G.C. (5)
808D3	GARBSCRATCH2	Saves counter in G.C. (5)
80920	GCOLCOUNT	Graphics #Cols (2)
8030E	GraphPrtHook	(11) aka: IRAMBEND
90000	HARDRAMEND	IRAM Home ends at #7FFFF Appears to be an obsolete constant from the 48G, where IRAM was only 32kB big and thus ranged from #80000 to #8FFFF. The description even seems to come from the 48S!
8075C	HISTORY1	-> \$ with the most recent CMD history entry
80761	HISTORY2	->2nd entry <see>HISTORY1
80766	HISTORY3	->3rd entry <see>HISTORY1
8076B	HISTORY4	->4th (oldest) entry <see>HISTORY1
8087A	HISTORYLEVEL	which stack level is next (1)
8000A	HOMEMASK	Home Size of RAM (mask) (5)
8000F	HRAMEND	M.S.N. of size of RAM chip (1)
8065A	INITEN	Warmstart Enable flag (1)
80669	INPUTSTREAM	Key Buffer (max 15 keys). (34) aka: KEYBUFFER
80523	IOCNIB	Saves IOC in OUTUART (1)
80927	IOCsave	Save of IOC before change (1)
80654	IOSAVE	Saves HiNib of ANNCTRL (1)
00219	IRAMBSIZE	Size of <see>IRAMBUFF
800F5	IRAMBUFF	Exec Buff (code under IRAM) (537)
80127	IRAMBUFF2	<see>IRAMBUFF +50
80005	IRAMMASK	IRAM Size Config Mask (5)
8064F	IREG	Saves Interrupt History (3)
80817	ITEM1LINES	# display lines currently (1)
80757	ITEM1STATE	->list of lists describing stack level 1
80775	KERMERRM	->Kermit error message aka: PDCSYMB
808ED	KERMMODE	Kermit Mode information (1)

8090C	KEYLIST	(5)
80911	KEYLOCK	(1)
8065C	KEYSTATE	location of kbd state (16)
807DE	KeyOb	->Pending key-object
80951	KeyRomPtr0	RomPtr for KeyOb (11)
8095C	KeyRomPtr1	RomPtr for MenuKey 1 (11)
80967	KeyRomPtr2	RomPtr for MenuKey 2 (11)
80972	KeyRomPtr3	RomPtr for MenuKey 3 (11)
8097D	KeyRomPtr4	RomPtr for MenuKey 4 (11)
80988	KeyRomPtr5	RomPtr for MenuKey 5 (11)
80993	KeyRomPtr6	RomPtr for MenuKey 6 (11)
80739	LASTARG	->1st argument saved in CK<n> aka: LASTARG1
8073E	LASTARG2	->2nd <see>LASTARG
80743	LASTARG3	->3rd <see>LASTARG
80748	LASTARG4	->4th <see>LASTARG
8074D	LASTARG5	->5th <see>LASTARG
8087B	LASTARGCOUNT	# of args saved by CK<n> (1)
8087C	LASTARGf	Flag #Args>3 (1)
8087D	LASTERROR	Save area for error number (5)
80879	LASTLAST?	true if lastkey was last hit (1)
808FB	LASTOP	3-state encoding of operand/ unary/binary (1)
807D9	LASTROMWDOB	->Last user-level ROM-WORD evaluated (set by CK<n>)
808FC	LEFTTREE	(3)
8069A	LINECOUNTg	Ghost for LINECOUNT (2)
80692	LINENIBSg	Ghost for LINENIBS (3)
80840	LPD_HIST	Low Power Detect History (1)
807B1	LabelDef	->How to make menu labels
807CA	LastContext	->RRP saved for CheckContext
807A2	LastMenuDef	->Last menu definition
8099E	LastMenuRow	(5)
8092C	LastPrntTime	Time (Upper 11 nibs) (11)
80928	LineByteCt	Line Byte Counter (2)
80077	LoBatTime	Flag periodic ((*)) updates (1)
808C5	MENULEVEL	User-menu level (5)
807A7	MenuData	->Menu data for touch table
8079D	MenuDef	->Current menu definition

807D4	MenuExitAct	->Menu exit action definition
807BB	MenuKeyLS	->Left-shift menu key handler
807B6	MenuKeyNS	->No-shift menu key handler
807C0	MenuKeyRS	->Right-shift menu key handler
80947	MenuRow	(5)
807AC	MenuRowAct	->Prev/Next action definition
80058	NEXTIRQ	Time at next Timer2 int. (13)
80835	NOALARMSRV	Flags Disable Alarm Service (1)
808F5	NODECOUNT	expr-tree node count (3)
808F8	OBTREELEN	object length (3)
808CA	OLDMENU	Saves previous menu number (2)
80642	SAVE_OR	aka: ORghost
80913	OB/EXP?	
80734	OSAVE	
808CD	PADCOUNT	Indentation count for decomp (1)
808E2	PADJSAVE1	Status save in PTRADJUST (1)
808E3	PADJSAVE2	RSTK save in PTRADJUST (10)
8077A	PAINTTREE	->hxs of "textbook-mode" graphics
80917	PARENCOUNT	(2)
80902	PARENTTREE	(3)
80833	PASTDUE	Flags Past Due Alarm (1)
8083B	PASTDUE#	
80770	PDCHXS	->hxs map of outermost symbolic
80937	PFIFO	FIFO Buffer (16)
8068B	POPPEDKEY	Last Key from POPKEY (2)
80536	PORTOEOS	(5)
8053B	PORT1EOS	(5)
80540	PORT2EOS	(5)
80905	PRECSTACK	Op Precedence textbook entry (7)
800E2	Port1CRC	CRC for Device in Port1 (4)
800E1	PortStat	Copy of CARDSTAT Nib (1)
80924	PrtStatus	CPU Status Bits et al. (3)
807E8	RAMEND	->End of RAM aka: SYSNOUPSTART
808FF	RIGHTTREE	(3)
80822	RNSEED	Random number seed (15)
80716	ROMPARTS	->RomParts Area
807C5	ReviewKey	->Review-key definition

80652	SEMAPH	Saves control byte for IREG (2)
8086F	STACKNUM	ref. number of 1st visible (5)
80720	STOPSIGN	(5)
80919	STRETCHCOUNT	(2)
800D4	SW_ETime	Stopwatch Elapsed Time Ticks (13)
800BE	SW_Image	"HH:MM:SS:ss" Stopwatch (22)
80078	StartTime	SelfTest Start Time (Ticks) (13)
808CC	T1COUNT	Decrementd by srvc_timer1 (1)
80702	TEMPENV	->LAM environments (5)
80092	TESTMSG	SelfTest Msg Buffer (44)
80065	TIMECRC	CRC CheckSum for NEXTIRQ (4)
80069	TIMExmit	Time at scheduled timeout (13) aka: TIMEOUT
80874	TOPLINE	Editline-segment which appears first on the screen (5)
8070C	TOUCHTAB	(5)
807CF	TrackAct	->Action when CONTEXT changes
80725	UserKeys	->User key assignments (5)
80818	VIEWLEVEL	stack element currently viewed (5)
8091C	XmitSrcvTOut	XMIT/SRECV timeout (2)
80752	leeway	->hxs which will be GC'ed in a very-low-memory condition

9 Miscellaneous Entries

9.1 Undescribed Entry Points

1EFD2	xDER
1F640	xFCNAPPLY
1F223	xINTEGRAL
1F3F3	xWHERE
560ED	xssgeneral
50E59	!#1+IF<dim-1
50EA5	!#1-IF>0
2B789	1/X15
648BD	>LASTRAM-WORD
715B1	?ACCPTR>
39BF3	?RollUpDA2
39FB0	AbbrevStack?
2B7DC	ADDF
1265A	addrADISP
4226A	addrATTNFLAG
0E7D3	addrClkOnNib
00D48	addrKEYSTATE
136AC	addrLINECNTg
0188D	addrORghost
04E66	addrTEMPENV
179E8	addrTEMPTOP
1263A	addrVDISP
1264A	addrVDISP2
1605F	addtics
42EC7	AdjEdModes
047CF	adrDISABLE_K
32CB6	adrGraphPrth
047DD	adrKEYBUFFER
42284	adrTIMEOUTCLK
312DA	adr_uart_han
2B770	aH>HMS
42113	ALARMxcp
3981B	AlgEntryStat

1568F	ALGeq?
001FF	allkeys
2B67D	aMODF
39673	AngleStatus
47984	APPprompt1!
479A7	APPprompt2
420F5	ATTNxcp
70601	BANKMTHDS
44F42	BindMatVars
008E6	BITMAP
2E108	BUILDKPACKET
27D00	check_pdata
32CAF	ChkGrHook
49C54	CkEQUtil
40882	(CkSecoType)
0D9C7	CKTIME
0CD3F	(CLKADJ*)
0D7A1	CLKUTL1
39FD2	ClrAbbrevStk
0ED78	(ClrDouseAlm)
2BBE2	CLRFRC
423D3	clrtypeout
5573D	COLAthexFCN
01FD3	Coldstart
09B73	COMPCONFRC
396C8	ComVecStatus
256E4	convertbase
4651C	CopyColsLeft
4677E	CopyColsRight
46625	CopyRowsDown
46409	CopyRowsUp
7DF87	COPYVAR
137DC	corner
15941	CRUNCHNoBlame
4248E	CtlAlarm!
424A1	(CtlAlarm@)
4E442	CURRENTMARK?
427AF	Cursor&Disp

13F01	(CURSOR+)
53A05	(D0=ALoop)
39371	(DA2bTemp?)
25223	DaDGNTc
0D4AD	DAY#
0D744	Day>Date
424DA	DCursor
2512D	delimcase
7DC54	derprod1
7DC0E	derquot
13B51	DISPCHAR+PC
133AB	disprange
153FC	DispVarsUtil
2BBB5	DIV2
42475	DoCAAlarmKey
3ADED	DoPlotMenu
31EE2	DOPRLCD
0DB51	dowutil
4C639	drax
2DDC4	DropSysErr\$
18308	DropSysObs
641CC	DupAndThen
00003	DZP
7DC88	easyabs
3EE47	Echo2Macros
039EF	ECUSER
44730	EDITPARTS
00019	ENTERCODE
4E46A	EQCURSOR?
32B08	ErrFixEIRU
0CBC4	ErrTime
1583C	EVALCRUNCH
4CF68	ExitFcn
4CE4C	EXITFCNsto
25C41	Extobcode
49BA5	FcnUtilEnd
32B1A	FixEIRU
17ADB	FixRRP

0D6D8	FLOAT
2BFFD	GETAB0
2BFE3	GETAB1
0D809	getBPOFF
21922	GetBVars
2C031	GETCDO
45D1F	GetElt
40BDD	(GetLastEdit)
45AE0	GetMat/Vec
138EF	GETPIX
138F2	GETPIX3
514AF	GETRHS
012EE	GetTimChk
0130E	GetTime++
0C50B0	~gFldVal
4CEE7	GraphicExit
1518D	GsstFIN
53860	HISTON?
0970A	InitEnab
45023	InitOldMat
385E8	InitSysUI
00110	IOC
0011F	IRAM@
0011A	IRC
3E5CD	IStackKey
426F1	LastERow?
50D78	LASTPT?
6515C	(lbrac)
4256B	LCursor
4E37E	LINECHANGE
39F6F	LINESOFSTACK
24C0D	List
15E83	longhxs
2D564	Loop
62ABB	MACRODCMP
7DF0E	MANMENU*/
7DF03	MANMENU+-
7DF66	MANMENUATG

7DF3A	MANMENUCSIV
7DF50	MANMENUCX
7DF45	MANMENUEQ
7DF24	MANMENUEXP
7DF2F	MANMENULN
7DF5B	MANMENUTRG
7DF19	MANMENU^
643F9	matchob?Lp
0120E4	~MESRclEqn
0D8AE	mpop1%
7DEED	nCOLCTQUOTE
255FB	need'case
25632	newBASE
4C09B	NEWINDEP
4E4B0	NEWMARK
6443A	nextpos
179D0	NEXTRRPOB
4BFAE	NEXTSTEP
29A8D	nextsym'R
255BD	ngsizecase
7DEA0	nINTGACOS
7DECC	nINTGALOG
7DE95	nINTGASIN
7DEAB	nINTGATAN
7DE5E	nINTGCOS
7DE7F	nINTGCOSH
7DED7	nINTGEXPM
7DE27	nINTGINV
7DEB6	nINTGLN
7DEC1	nINTGLOG
7DE32	nINTGSIGN
7DE53	nINTGSIN
7DE74	nINTGSINH
7DE48	nINTGSQ
7DE3D	nINTGSQRT
7DE69	nINTGTAN
7DE8A	nINTGTANH
53AE4	NoIgnoreAlm

01FDA	norecCSseq
3303F	NUMSOLVE
7DBAB	nWHEREEDER
7DBD7	nWHEREFCNAPP
7DBA0	nWHEREIFTE
7DBB6	nWHEREINTG
7DBC1	nWHEREESUM
7DBCC	nWHEREWHERE
216D8	OB>BAKcode
6207D	OverWrF/TLp
62B1F	PALPTRDCMP
39971	PathStatus
0630E3	~PCunpack
138CA	pixon2
49AD3	PointDerivUt
49F06	PointMoveCur
38B45	POLErrorTrap
0D92C	POPDATE%
0D948	POPTIME%
1BB41	preFACT
4E497	PREMARKON
39853	PrgmEntrStat
028FC	PRLG
323E9	PSubErr
5133C	PtoR
7DF71	PTYPE>PINFO
2C04B	PUTAB0
30E4E	PutSerialEck
4AB2A	PvarsC%0
00114	RBR
01AD7	RCKBp
00111	RCS
265ED	realPAcode
510D5	RECORDX&YC%
53A20	REPLACE_MODE
0C147	restoreiram
2BEEC	RNDC[B]
61FCF	Rom-Word?

49BD2	RootUtil	
45C2F	RowElt#	
4489E	ROWNUM	
1B185	rpnXROOT	
71DB2	RPTRACC	
0A532	SAFESKIPOB	
0000F	sALLOWINTR	
61D3A	SAVELAM	
01307	SavPtrTime*	
00008	sBEG	
00004	sBPOFF	
07661	SET	
39FC1	SetAbbrevStk	
38CDF	(SetBadPOLUI)	
53B31	setflag	
539F9	SetISysFlag	
4CF41	SETLOOPENV	
21CBA	SETROMPART	
423BB	settimeout	
00001	Sfkey1	
00006	Sfkey6	
0D50B0	~sFldVal	
42131	SLEEPxcp	
00002	sNEGATE	aka: sFLUSH
7DEF8	SPLITWHERE	
2BA0F	SQRF	
0131D	svrc_timer2	
1686A	stackitw	
3858E	StartupProc	
47467	STOAPPLDATA	
00001	sTRUNC	
29BC2	subpdcptch	
5A310	symbn	
32FF9	SYMBNUMSOLVE	
38728	SysErrorTrap	
08D66	SysPtr@	
40792	SystemLevel?	
00116	TBR	

00112	TCS	
4227F	TIMEOUT?	
0012E	TIMERCTRL.1	
0012F	TIMERCTRL.2	
3E586	TogInsertKey	
4272D	TopERow?	
2BD76	TST15	
3A9CE	TurnOffKey	
42660	UCursor	
39748	UserFlagStat	
397BB	UserKeysStat	
0580E7	~UTTYPEEXT0?	
0110E7	~UTVUNS1Arg	
17B86	VLM	
31416	(WaitTbz0)	
01FBD	Warmstart	
		aka: norecPWLseq
136AA	WindowXY	
0931B	X@	
2BD32	Y<=X	
0DB3A	YMD>Ticks	
4E776	Z-BOX	

10 Entries sorted by address

Here follows a list of entries sorted by address. Six-digit addresses are always sorted after five-digit addresses. The six-digit addresses for rompointers and flashpointers consist of the pointer number (first three digits) and the flashbank/library id (last three digits). Sorting of these addresses uses first the flashbank/library id and then the pointer number, so 000123 will be sorted after FFF122.

00001	Ann_RAD.b	00104	CRC	0012E	TIMERCTRL.1
00001	sTRUNC	00105	COMMANDMASK	0012F	TIMER2CTRL
00001	Sfkey1	00106	Do1UserMASK	0012F	TIMERCTRL.2
00001	mEditLExists	00107	DA1ValidMASK	00130	MENUADDR
00001	ParenModmask	00108	DA1NoChMASK	00137	TIMER1
00002	sFLUSH	00108	POWERSTATUS	00138	TIMER2
00002	sNEGATE	00109	POWERCTRL	001CE	NOP1MASK15
00002	Ann_IO.b	00109	DA1BadMASK	001CE	BadTOLUIMASK
00003	DZP	0010A	EDITLMASK	001FF	allkeys
00004	Ann_Susp.b	0010B	ParenModMASK	00201	AbbrStkMASK
00004	sBPOFF	0010B	ANNCTRL	00202	argtypeerr
00006	Sfkey6	0010C	ANNCTRL2	00202	ServModeMASK
00008	IRAMHOMEmsn	0010C	DispTimeMASK	00203	INSERTMASK
00008	Ann_Alert.b	0010D	NOP1MASK13	00203	argvalerr
00008	sBEG	0010E	NOP1MASK14	00204	DA2bTempMASK
0000A	portnotaverr	00110	NOP1MASK16	00205	BLINKMASK
0000F	sALLOWINTR	00110	IOC	00206	ASuspOKMASK
00010	Ann_Shift.b	00111	NOP1MASK17	00207	DA2aValdMASK
00011	LOCUPSIZE	00111	RCS	00208	DA2aNoChMASK
00019	ENTERCODE	00112	TCS	00209	DA2aBadMASK
00020	Ann_ALT.b	00112	NOP1MASK18	0020A	NAppKeyMASK
00025	OBUPSIZE	00113	CRER	0020B	1PDCMASK
00036	SYSUPSIZE	00113	NOP1MASK19	0020C	NOP2MASK12
00040	Ann_Alpha.b	00114	NOP1MASK20	0020D	NOP2MASK13
00080	Ann_Busy.b	00114	RBR	0020E	NOP2MASK14
00100	DISPIO	00116	TBR	00210	NOP2MASK16
00101	NoRo1DA2MASK	0011A	IRC	00211	NOP2MASK17
00101	CONTRAST	0011F	IRAM@	00212	NOP2MASK18
00102	ReqClkOnMASK	00120	DISPADDR	00213	NOP2MASK19
00102	DISPTEST	00125	LINEOFFS	00214	NOP2MASK20
00103	UNDOMASK	00128	LINECOUNT	00219	IRAMBSIZE
00104	DA2aTempMASK	0012E	TIMER1CTRL	002CE	NOP2MASK15

002CE	NoAlgProcess	0080C	NOP8MASK12	018E2	clkspd
00301	posunferr	0080D	NOP8MASK13	01AD7	RCKBp
00302	negunferr	0080E	NOP8MASK14	01B8F	DispOn
00303	ofloerr	00810	NOP8MASK16	01BBD	DispOff
00305	infreserr	00811	NOP8MASK17	01C31	DO->Row1
00401	DA2bIsEdMASK	00812	NOP8MASK18	01C58	DO->Sft1
00402	TrackMASK	00813	NOP8MASK19	01F6D	CLCD10
00403	ALGMASK	00814	NOP8MASK20	01FA7	CLEARLCD
00404	DA3TempMASK	008CE	NOP8MASK15	01FBD	norecPWLseq
00405	LOWERMASK	008CE	DoCreateMenu	01FBD	Warmstart
00406	BadPOLUIMASK	008E6	BITMAP	01FD3	Coldstart
00407	DA2bValdMASK	009A5	Debounce	01FDA	norecCSseq
00408	DA2bNoChMASK	00A03	intrptderr	0210F	DOROMPOLL
00409	DA2bBadMASK	00B02	constuniterr	021DD	ROMPOLL
0040A	NUsrKeyMASK	00C02	timeouterr	028FC	PRLG
0040B	NewEditLMASK	00C06	xferfailerr	02911	DOBINT
0040C	NOP4MASK12	00C0D	kermsendmsg	02933	DOREAL
0040D	NOP4MASK13	00C0E	kermrecvmsg	02955	DOEREL
0040E	NOP4MASK14	00C10	kermpktmsg	02977	DOCMP
00410	NOP4MASK16	00C13	prtparerr	0299D	DOECMP
00411	NOP4MASK17	00C14	lowbaterr	029BF	DOCHAR
00412	NOP4MASK18	00C74	OnKeyDown?	029E8	DOARRY
00413	NOP4MASK19	00C80	OnKeyStable?	02A0A	DOLNKARRY
00414	NOP4MASK20	00D48	addrKEYSTATE	02A2C	DOCSTR
004CE	NOP4MASK15	00D57	Flush	02A4E	DOHSTR
004CE	InSimplyExpr	00D71	FLUSHKEYS	02A4E	DOHXS
007B5	SrvcKbdAB	00D71	FLUSH	02A74	DOLIST
00801	IgnorAlmMASK	00D8E	FlushAttn	02A96	DORRP
00802	BadMenuMASK	010E5	AllowIntr	02AB8	DOSYMB
00803	PRINTINGMASK	01115	DisableIntr	02ADA	DOEXT
00804	RebuildMASK	0115A	AINRTN	02AFC	DOTAG
00805	STKDCMASK	01160	CINRTN	02B1E	DOGROB
00806	DA1TempMASK	012EE	GetTimChk	02B40	DOLIB
00807	DA3ValidMASK	01307	SavPtrTime*	02B62	DOBAK
00808	DA3NoChMASK	0130E	GetTime++	02B88	DOEXT0
00809	DA3BadMASK	0131D	svrc_timer2	02BAA	DOEXT1
0080A	AppModeMASK	017A6	makebeep	02BAA	DOACPTR
0080B	DoStdKeyMASK	0188D	addrORghost	02BCC	DOEXT2

02BEE	DOEXT3	039EF	ECUSER	03FC7	TYPERRP
02C10	DOEXT4	03A81	TRUE	03FD1	TYPELAM
02D9D	DOCOL	03A86	PUSHA	03FDB	TYPEEREL
02DCC	DOCODE	03AC0	FALSE	03FE5	TYPEEXT
02E48	DOIDNT	03ADA	XOR	03FEF	any
02E6D	DOLAM	03AF2	NOT	03FEF	ZERO
02E92	DOROMP	03B2E	EQ	03FEF	BINT0
02FD6	DoLam	03B46	AND	03FF9	real
02FEF	ROMSEC	03B75	OR	03FF9	BINT1
03019	SKIPOB	03B97	EQUAL	03FF9	MEMERR
0312B	SEMI	03C64	TYPE	03FF9	ONE
0314C	DEPTH	03CA6	#0=	04003	BINT2
03188	DUP	03CC7	#0<>	04003	TWO
031AC	2DUP	03CE4	#<	04003	cmp
031D9	NDUP	03D19	#=	0400D	str
03223	SWAP	03D4E	#<>	0400D	BINT3
03244	DROP	03D83	#>	0400D	THREE
03249	DropLoop	03DBC	#+	04017	FOUR
03258	2DROP	03DC7	Push#Loop	04017	BINT4
0326E	NDROP	03DE0	#-	04017	arry
03295	ROT	03DEF	#1+	04021	FIVE
032C2	OVER	03E0E	#1-	04021	BINT5
032E2	PICK	03E2D	#2+	04021	list
03325	ROLL	03E4E	#2-	0402B	id
0339E	UNROLL	03E6F	#2*	0402B	BINT6
03442	MAKEARRY	03E8E	#2/	0402B	SIX
03562	ARSIZE	03EB1	#AND	0402B	idnt
0357C	PUSH#ALoop	03EC2	##	04035	lam
0357F	PUSH#Lloop	03EF7	#/	04035	BINT7
035A9	DIMLIMITS	03F14	Push2#Loop	04035	SEVEN
0366F	GPOverWrROLp	03F24	IntDiv	0403F	seco
03672	GPOverWrALp	03F5D	POP2#	0403F	EIGHT
03685	ARRYEL?	03F8B	TYPEREAL	0403F	BINT8
03685	FINDELN	03F95	TYPECMP	04049	symb
036F7	Push#TLoop	03F9F	TYPELIST	04049	BINT9
0371D	GETATELN	03FA9	TYPEIDNT	04049	NINE
03991	MUL#	03FB3	TYPECOL	04053	sym
039BE	GETTEMP	03FBD	TYPESYMB	04053	BINT10

04053	TEN	040E9	BINT25	04193	FORTYTWO
0405D	hxs	040E9	TWENTYFIVE	0419D	BINT43
0405D	BINT11	040F3	REALSYM	0419D	FORTYTHREE
0405D	ELEVEN	040F3	BINT26	041A7	TurnOff
04067	BINT12	040F3	TWENTYSIX	041DA	!TurnOff
04067	grob	040FD	TWENTYSEVEN	041ED	DEEPSLEEP
04067	TWELVE	040FD	BINT27	0426A	ShowInvRomp
04071	TAGGED	04107	TWENTYEIGHT	04292	DeepSleep
04071	BINT13	04107	BINT28	04544	AlertStatus
04071	THIRTEEN	04111	BINT29	04575	Alert\$
0407B	EXT	04111	TWENTYNINE	04708	CHECKKEY
0407B	BINT14	0411B	BINT30	04714	GETTOUCH
0407B	FOURTEEN	0411B	REALEXT	047C7	REPKEY?
0407B	unitob	0411B	THIRTY	047CF	adrDISABLE_K
04085	BINT15	04125	THIRTYONE	047DD	adrKEYBUFFER
04085	rompointer	04125	BINT31	04840	POPKEY
04085	FIFTEEN	0412F	BINT32	04912	LiteSlp
0408F	REALOB	0412F	THIRTYTWO	04929	liteslp
0408F	BINT16	04139	BINT33	04988	Attn?
0408F	SIXTEEN	04139	THIRTYTHREE	04999	KeyInBuff?
04099	SEVENTEEN	04143	THIRTYFOUR	04A0B	GETPROC
04099	2REAL	04143	BINT34	04A41	GETDF
04099	REALREAL	0414D	BINT35	04CE6	ERROR@
04099	BINT17	0414D	THIRTYFIVE	04DOE	ERRORSTO
040A3	EIGHTEEN	04157	TTHIRTYSIX	04D33	ERRORCLR
040A3	BINT18	04157	BINT36	04D3E	DROPNULL\$
040AD	NINETEEN	04161	THIRTYSEVEN	04D57	TWODROPNULL\$
040AD	BINT19	04161	BINT37	04D64	GETTHEMESG
040B7	BINT20	0416B	THIRTYEIGHT	04D87	JstGETTHEMESG
040B7	TWENTY	0416B	BINT38	04D87	JstGetTHEMESG
040C1	TWENTYONE	04175	BINT39	04DD7	SPLITmsg
040C1	BINT21	04175	THIRTYNINE	04E07	GETEXITMSG
040CB	BINT22	0417F	FOURTY	04E37	EXITMSGSTO
040CB	TWENTYTWO	0417F	BINT40	04E5E	ERRSET
040D5	BINT23	0417F	FORTY	04E66	addrTEMPENV
040D5	TWENTYTHREE	04189	FORTYONE	04EA4	ABORT
040DF	BINT24	04189	BINT41	04EB8	ERRTRAP
040DF	TWENTYFOUR	04193	BINT42	04ED1	ERRJMP

04FAA	SETLBERR	055E9	NULL{}	0613E	GARBAGECOL
04FB6	SETMEMERR	055F3	NULLSYMB	064BD	TOTEMPOBADJ
04FBB	DOMEMERR	055FD	NULL::	064D6	DOADJ1
04FC2	SETDIRRECUR	05616	LENHXS	064E2	DOADJ
04FCE	SETLAMERR	05622	OVERLEN\$	06529	PUSH2#
04FDA	SETCORPORT	05636	LEN\$	06537	PUSH#
04FE6	SETOBINUSE	0567B	LENCOMP	065AA	GPMEMERR
04FF2	SETPORTNOTAV	056B6	NTHELCOMP	065D9	PTRREFD?
04FFE	SETNOROOM	05733	SUB\$	065E5	REFERENCED?
0500A	SETXNONEXT	05815	SUBHXS	06641	POP#
05016	SETROMPERR	05821	SUBCOMP	06657	TOTEMPOB
05023	Errjmp	05902	OSIZE	066B9	MOVEUP
05040	ATTNFLAG	05944	OCRC	0670C	MOVEDOWN
05068	ATTNFLAGCLR	0596D	PUSHhxsLoop	0675C	WIPEOUT
05089	CARCOMP	0597E	DoCRcC	0679B	SAVPTR
050ED	CAR\$	05981	DoCRC	067D2	GETPTR
05143	GETPTRLOOP	059CC	#>HXS	06806	ROOM
05153	CDRCOMP	05A03	HXS>#	06992	MOVERSD
0516C	CDR\$	05A51	CHR>#	069C5	MOVEDSU
0518A	&HXS	05A75	#>CHR	069F7	ADJMEM
05193	&\$	05AB3	CHANGETYPE	06A1D	MOVEDSD
0521F	&COMP	05ACC	!CHANGETYPE	06A53	MOVERSU
0525B	>H\$	05AED	ID>LAM	06A8E	DIV5
052C6	>HCOMP	05B01	LAM>ID	06AD8	CREATETEMP
052EE	>T\$	05B15	\$>ID	06B3E	FREEINTEMP?
052FA	>TCOMP	05B79	MAKE\$	06B4E	INTEMNOTREF?
05331	COMP	05B7D	MAKE\$N	06BC2	NOTREF?
05445	::N	05BE9	ID>\$	06DDE	>TOPTEMP
05459	{ }N	05C27	%>C%	06E8E	NOP
0546D	SYMBN	05C72	%>C%>	06E97	'
05481	EXTN	05D2C	C%>%	06EEB	'R
054AF	INNERCOMP	05DBC	C%>%>%	06F66	'REVAL
0554C	DOGARBAGE	05E81	>TAG	06F8E	EVAL
05566	NULLHXS?	05E9F	{ }>TAG	06F9F	>R
0556F	NULL\$?	05EC9	TAG>	06FB7	RDROP
055B7	NULLCOMP?	05F2E	ID>TAG	06FD1	COLA
055D5	NULLHXS	05F42	GARBAGE	07012	R@
055DF	NULL\$	05F61	MEM	0701F	R>

070C3	RPITE	077E4	MAKERRP	08D92	SYSCONTEXT
070FD	RPIT	078E9	FIRST@LAM	08D92	HOMEDIR
0712A	?SKIP	078F5	NTH@LAM	08DC4	SYSSTOPSIGN
0712A	NOT_IT	07943	@LAM	08DD4	SYSRRP?
0714D	SKIP	0797B	@	08DF2	CREATERRP
0715C	2SKIP	07C18	COMPILEID	0931B	X@
0716B	IDUP	07D1B	STOLAM	0948E	BAK>OB
071A2	BEGIN	07D27	STO	0970A	InitEnab
071AB	AGAIN	07E50	#>ROMPTR	09B73	COMPCONFRC
071C8	UNTIL	07E76	PTR>ROMPTR	0A00E	CKLBCRC
071E5	REPEAT	07E99	ROMPTR@	0A532	SAFESKIPOB
071EE	WHILE	07F86	ROMPART	0AAB2	PORTSTATUS
07221	INDEX@	08081	ROMPART>ADDR	0AB22	PORTEND
07249	ISTOP@	080BF	ROMPARTSIZE	0AB82	NEXTLIBBAK
07258	JINDEX@	080DA	NEXTROMPID	0B409	MERGE
07264	JSTOP@	08112	GETHASH	0BBED	TrueTrue
07270	INDEXSTO	08130	GETMSG	0BC01	failed
07295	ISTOPSTO	0813C	GETLINK	0BC6F	!*trior
072AD	JINDEXSTO	08157	GETCONFIG	0BCCF	!*triand
072C2	JSTOPSTO	08199	ROMPARTNAME	0BD54	tok8cktrior
07321	STOPLOOP	081D9	BAKNAME	0BD60	tok8trior
07334	LOOP	081DE	LIB>#	0C147	restoreiram
073A5	+LOOP	081E3	PTR>ID	0C501	GETEL
073C3	ZERO_DO	081FB	ROMPTRDECOMP	0C506	rGETATELN
073CE	ONE_DO	082E3	RAM-WORDNAME	0CA60	ATTNchk
073DB	#1+_ONE_DO	08309	MYRAMROMPAIR	0CBAE	NOALARMERR
073F7	DO	08326	LASTRAM-WORD	0CBC4	ErrTime
07497	ABND	08376	PREVRAM-WORD	0CBFA	TOD
074D0	BIND	083D1	GETRRP	0CC0E	DATE
074E4	DOBIND	085D3	REPLACE	0CC39	DDAYS
075A5	GETLAM	08696	CREATE	0CC5B	DATE+DAYS
075E9	PUTLAM	08C27	PURGE	0CD2B	>DATE
07638	SETHASH	08CCC	ROMPTR>#	0CD3F	CLKADJ
0764E	SETMSG	08D08	CONTEXT!	0CD3F	CLKADJ*
07661	SET	08D4A	STOPSIGN!	0CD53	>TIME
076AE	OFFRRP	08D5A	CONTEXT@	0CF5B	Ticks>wd\$
07709	TOSRRP	08D66	SysPtr@	0CFD9	Date>d\$
0778D	ONSRRP?	08D82	STOPSIGN@	0D06A	TOD>t\$

0D143	Ticks>TOD	0E235	ALARMS@	0F615	UMCHS
0D156	Ticks>Date	0E3DF	RCLALARM%	0F62E	UMSIN
0D169	Ticks>Rpt	0E402	RCLALM	0F660	UMCOS
0D18A	WSLOGN	0E461	INSERTN{}	0F674	UMTAN
0D2A3	WSLOG	0E4DE	REMOVEN{}	0F688	%%2PI
0D2F0	Date>wd\$	0E510	STOALARM%	0F6A2	UM+
0D304	TIMESTR	0E54D	STOALARMLS	0F774	UM-
0D4AD	DAY#	0E66A	VerifyTOD	0F792	UM*
0D5E5	ASRW5	0E6ED	STOALM	0F823	UM/
0D5F6	ASLW5	0E724	PURGALARM%	0F841	UMINV
0D607	CSRW5	0E724	DELALARM	0F8FA	UMXROOT
0D618	CSLW5	0E7D3	addrClkOnNib	0F913	UMSQ
0D62F	DCHXW	0EAD7	FINDALARM%	0F92C	UMSQRT
0D6D8	FLOAT	0EB31	FNDALARM{}	0F945	UMSI
0D744	Day>Date	0EB31	FINDALARMLS	0FA69	tok_g
0D7A1	CLKUTL1	0EB81	SysTime	0FA8E	tok_m
0D809	getBPOFF	0EB81	CLKTICKS	0FACE	tok_s
0D8AE	mpop1%	0EBD5	FindNext	0FB6F	UMMAX
0D92C	POPDATE%	0ED78	ClrDouseAlm	0FB8D	UMMIN
0D948	POPTIME%	0EDE1	MAKEHXS	0FBAB	UM%
0D9C7	CKTIME	0EE26	Date+Time	0FC3C	UM%CH
0DB3A	YMD>Ticks	0EE50	Date>hxs13	0FCCD	UM%T
0DB51	dowutil	0EE83	TOD>Ticks	0FCE6	UMSIGN
0DB91	HXDCW	0EF45	>ALRMLS	0FCFA	UMIP
0DDA8	ACKALLALMS	0F02D	%TICKSday	0FDOE	UMFP
0DDC1	ACKALM	0F218	UNIT>\$	0FD22	UMFLOOR
0E029	DISPROW1*!	0F33A	UM>U	0FD36	UMCEIL
0E047	Save16	0F34E	UMU>	0FD4A	UMOPER:
0E05B	Restore16	0F371	UMCONV	0FD68	UMRND
0E06F	Clr16	0F3E4	puretemp?	0FD8B	UMTRC
0E083	Clr8	0F41B	TempConv	0FE44	U>NCQ
0E097	Clr8-15	0F584	UM=?	10029	4DROPTRUE
0E0AB	TOP16	0F598	UM#?	10047	U>nbr
0E0D3	TOP8	0F5AC	UM<?	10065	Unbr>U
0E0FB	Rows8-15	0F5C0	UM>?	10ADB	UobROT
0E128	HBUFF_X_Y	0F5D4	UM<=?	10AF9	unrot1
0E128	HBUF_X_Y	0F5E8	UM>=?	10B5E	um*
0E1D8	ALRMLS>	0F5FC	UMABS	10B68	um/

10B72	um^	11462	FIRSTC+	124CB	DISPROW8
10B7C	umP	11501	Lock?	12635	HARDBUFF
10B86	umEND	11511	PrgmEntry?	1263A	addrVDISP
10E36	8NULLLAM{}	11533	SetPrgmEntry	12645	HARDBUFF2
10E68	cfF	11543	SetLock	1264A	addrVDISP2
10E82	cfC	1155C	ClrPrgmEntry	12655	ABUFF
10E9C	%%KZERO	1156C	ClrLock	1265A	addrADISP
10EB6	%%RZERO	1158F	MAKEGROB	12665	GBUFF
10EEA	INVUNITERR	115B3	makegrob	126DF	BLANKIT
10EFA	CONSTUNITERR	1165A	w->W	1270C	DISPSTATUS2
10F40	GPErrjmpC	11679	GROB!	12725	DISPROW1*
10F54	NULLCHARERR	116B5	grob!	12748	DISPROW2*
10F64	INVFUNCERR	1192F	SUBGROB	12770	COERCE\$22
10F74	NOEQERR	11A6D	GROB!ZERO	127A7	SEP\$NL
10F80	ErrjmpC	11CF3	\$>BIGGROB	127CA	DROPDUPLEN\$1+
10F86	SYNTAXERR	11D00	\$>GROB	128B0	XYGROBDISP
10F96	SETINVPPAR	11D8F	\$5x7	12964	HEIGHTENHBUFF
10FA6	SETNONERAL	11F80	\$>grob	1297D	BROADENHBUFF
10FB6	SETISOLERR	1200C	RIGHT\$3x6	12996	ScreenUpN
10FC6	NOHALTERR	1215E	CENTER\$3x5	12A0D	ScreenUp
10FE6	LASTSTKERR	122FF	INVGROB	12A4A	ScreenDnN
10FF6	LASTCMDERR	12429	DISPN	12AF6	ScreenDn
11006	ARGNUMERR	12429	BIGDISPN	12B58	HBUFFDIMw
11016	SETCIRCERR	1245B	DISPROW1	12B6C	HARDHEIGHT
11026	DIRARGERR	1245B	DISP@01	12B85	FlashMsg
11036	EMPTYDIRERR	1245B	BIGDISPROW1	12BB7	BROADENGROB
11046	INVDEFERR	1246B	DISPROW2	12DD1	HEIGHTENGROB
11056	MISLIBERR	1246B	DISP@09	12F94	GROB>GDISP
11066	IDCONFERR	1246B	BIGDISPROW2	13043	KILLADISP
11320	ClrBusyAnn	1247B	BIGDISPROW3	13061	KILLGDISP
1132D	SetAlphaAnn	1247B	DISPROW3	130AC	RECLAIMDISP
1133A	ClrAlphaAnn	1247B	DISP@17	130CA	RSZVDISP
11347	SetRightAnn	1248B	BIGDISPROW4	13135	TOGDISP
11354	ClrRightAnn	1248B	DISP@25	1314D	TOADISP
11361	SetLeftAnn	1248B	DISPROW4	13167	GDISPON?
1136E	ClrLeftAnn	1249B	DISPROW5	131C8	WINDOWUP
11432	FIRSTC@	124AB	DISPROW6	13220	WINDOWDOWN
11442	SETFIRSTC_0	124BB	DISPROW7	133AB	disprange

134AE	CLEARVDISP	14137	DOSTR>	15A60	rstfmt1
134E4	WINDOWLEFT	1415A	DOBEEP	15A8B	savefmt1
1357F	WINDOWRIGHT	141B2	setbeep	15B13	DECOMP\$
13679	WINDOWXY	141E5	ERRBEEP	15B31	editdecomp\$w
13695	REDISPHBUFF	1420A	\$>\$?	15D6F	BINT_117h
136AA	WindowXY	142A6	\$<\$?	15E83	longhxs
136AC	addrLINECNTg	142BA	\$>=\$?	15EF6	ONEPOS\$
137B6	WINDOWCORNER	142E2	\$<=\$?	1605F	addtics
137DC	corner	142FB	Ck&Freeze	1613F	NULL\$TEMP
1380F	PIXOFF3	1446F	SuspendOK?	162B8	a%>\$
13825	PIXON3	14483	nohalt	162B8	a%>\$,
1383B	PIXOFF	14C17	sstDISP	16671	Shrink\$
1384A	PIXON	14EA5	RDUP	16677	Clean\$R0
138CA	pixon2	14F2A	RROLL	16683	Clean\$
138EF	GETPIX	15007	DO%EXIT	166E3	DOFIX
138F2	GETPIX3	1501B	DOHXSEXIT	166EF	DOSCI
13986	PIXON?3	1502F	DO#EXIT	166FB	DOENG
13992	PIXON?	15048	DO\$EXIT	16707	DOSTD
13B51	DISPCHAR+PC	1518D	GsstFIN	167BF	DPRADIX?
13D28	cursorblink+	151A6	SolvMenuInit	167D8	#:>\$
13D55	cursorblink-	152FF	EqList?	167E4	#>\$
13D8C	CURSOR1	153FC	DispVarsUtil	1686A	stackitw
13DB4	CURSOR2	1553B	KeepUnit	169A5	NDROPFALSE
13E85	CURSOR_OFF	1568F	ALGeq?	1782E	2LEN\$#+
13EBC	CURSOR_OFF0	15717	DOSTOE	1795A	!DcompWidth
13ED2	CURSOR_OFF+	1572B	DORCLE	17980	DcompWidth@
13EF1	CURSOR@	15744	EQUATION	179D0	NEXTRRPOB
13F01	CURSOR+	1576C	CUREQ	179E8	addrTEMPTOP
13F29	SETCURSOR	15777	NULLID	17ADB	FixRRP
13FE5	SAVEERRN	1578D	CRUNCH	17B86	VLM
1400E	ERRO	1583C	EVALCRUNCH	18308	DropSysObs
14039	ERRN	1592D	CK1NoBlame	1848C	PATHDIR
1404C	ERRN>HXS	15941	CRUNCHNoBlame	184E1	CREATEDIR
14065	ERRM	15978	1stkdecomp\$w	18513	?STO_HERE
14088	DO>STR	159EB	stkdecomp\$w	18513	XEQSTOID
140AB	DODISP	159FA	setStdWid	1853B	SAFE@_HERE
140F1	DOCHR	15A0E	EDITDECOMP\$	1854F	?PURGE_HERE
1410F	DONUM	15A40	ederr	18595	XEQPGDIR

185C7	DoHere:	18DBF	UNCOERCE	197C8	UMFACT
18621	LastNonNull	18E45	DNOTSYMB?SEMI	197F7	xTIME
1863A	PrevNonNull	18EBA	COMPEVAL	19812	xDATE
18653	CkNonNull	18ECE	CK1&Dispatch	1982D	xTICKS
1867F	DODIRPRG	18EDF	CK2&Dispatch	19848	xWSLOG
186E8	DOTVARS%	18EF0	CK3&Dispatch	19863	xACKALL
18779	DOVARS	18F01	CK4&Dispatch	1987E	xACK
1884D	OLASTOWDOB!	18F12	CK5&Dispatch	1989E	xSETDATE
1884D	OLastRomWrd!	18F23	EvalNoCK	198BE	xSETTIME
18873	AND\$	18F6A	'EvalNoCK:_sup	198DE	xCLKADJ
18887	OR\$	18F6A	EvalNoCK:	198FE	xSTOALARM
1889B	XOR\$	18F9D	CK&DISPATCH0	19928	xRCLALARM
188D2	NOT\$	18FA9	CK&DISPATCH2	19948	xFINDALARM
188E6	!AND\$	18FB2	CK&DISPATCH1	19972	xDELALARM
188F5	!OR\$	191B9	##*OVF	19992	xTSTR
18904	!XOR\$	19207	CKNFLOATS	199B2	xDDAYS
18961	!NOT\$	19294	>ARRY	199D2	xDATE+
18A15	CKONOLASTWD	19350	NOTINHARDROM?	199EB	DoInputForm
18A1E	CK0	193DA	COERCE-{}2	1A105	xCRDIR
18A5B	CK3	194F7	COERCE2	1A125	xPATH
18A68	CK3NOLASTWD	1950B	UNCOERCE2	1A140	xHOME
18A80	CK2	19529	UNCOERCE-{}2	1A15B	xUPDIR
18A8D	CK2NOLASTWD	1957B	xASR	1A16F	UPDIR
18AA5	CK1	1959B	xRL	1A194	xVARS
18AB2	CK1NOLASTWD	195BB	xRLB	1A1AF	xTVARS
18B6D	CK5	195DB	xRR	1A1D9	xBYTES
18B7A	CK5NOLASTWD	195FB	xRRB	1A1FC	OCRC%
18B92	CK4	1961B	xSL	1A265	VARSIZE
18B9F	CK4NOLASTWD	1963B	xSLB	1A2BC	xNEWOB
18C34	CKN	1965B	xSR	1A2DA	INHARDROM?
18C4A	CKNNOLASTWD	1967B	xSRB	1A303	xKILL
18C92	SETNONEXTERR	1969B	xR>B	1A31E	xOFF
18CA2	SETSIZEERR	196BB	xB>R	1A339	xDOERR
18CA7	DOSIZEERR	196DB	xCONVERT	1A36D	xERRO
18CB2	SETTYPEERR	1971B	xUVAL	1A388	xERRN
18CC2	SETSTACKERR	1974F	x>UNIT	1A3A3	xERRM
18CD7	%ABSCOERCE	19771	xUBASE	1A3BE	xEVAL
18CEA	COERCE	197A5	xUFACT	1A3FE	xIFTE

1A4A3	%IFTE	1B374	xSQRT	1BD55	xRND
1A4CD	xIFT	1B3F5	CK%SQRT	1BDD1	xTRNC
1A52E	xSYSEVAL	1B426	xSQ	1BE4D	xMOD
1A584	xDISP	1B47B	%SQ	1BE9C	xMANT
1A5A4	xFREEZE	1B48F	C%C*C	1BEC8	xD>R
1A5C4	xBEEP	1B4AC	xSIN	1BEF4	xR>D
1A5E4	x>NUM	1B505	xCOS	1BF1E	x>HMS
1A604	xLAST	1B55E	xTAN	1BF3E	xHMS>
1A71F	xWAIT	1B5B7	xSINH	1BF5E	xHMS+
1A738	Wait/GetKey	1B606	xCOSH	1BF7E	xHMS-
1A7B5	dowait/quit?	1B655	xTANH	1BF9E	xRNRM
1A7C9	dowait	1B6A4	xASIN	1BFBE	xCNRM
1A7ED	wait	1B6EA	CK%ASIN	1BFDE	xDET
1A858	xCLLCD	1B72F	xACOS	1BFFE	xDOT
1A873	xKEY	1B775	CK%ACOS	1C01E	xCROSS
1A8BB	xCONT	1B79C	xATAN	1C03E	xRSD
1A8D8	x=	1B7EB	xASINH	1C060	x%
1A995	xNEG	1B830	xACOSH	1COD7	x%T
1AA1F	xABS	1B86C	CK%ACOSH	1C149	x%CH
1AA6E	xCONJ	1B8A2	xATANH	1C1B9	xRAND
1AABD	xPI	1B8DE	CK%ATANH	1C1D4	xRDZ
1AADF	xMAXR	1B905	xEXP	1C1F6	xCOMB
1AB01	xMINR	1B94F	xLN	1C236	xPERM
1AB23	xCONSTANTe	1B995	CK%LN	1C274	xSF
1AB45	xi	1B9C6	xLOG	1C2D5	xCF
1AB67	x+	1BA0C	CK%LOG	1C313	xFS?
1AC93	SWAP>HCOMP	1BA3D	xALOG	1C360	xFC?
1ACDD	xNEGNEG	1BA8C	xLNP1	1C399	xDEG
1AD09	x-	1BAC2	xEXPM	1C3B4	xRAD
1ADEE	x*	1BB02	xFACT	1C3CF	xGRAD
1AF05	x/	1BB41	preFACT	1C3EA	xFIX
1B02D	x^	1BB6D	xIP	1C41E	xSCI
1B185	rpnXROOT	1BBA3	xFP	1C452	xENG
1B1CA	xXROOT	1BBD9	xFLOOR	1C486	xSTD
1B278	xINV	1BC0F	xCEIL	1C4A1	xFS?C
1B2DB	xARG	1BC45	xXPON	1C4CE	TestUserClr
1B30D	%ARG	1BC71	xMAX	1C4EC	TestSysClr
1B32A	xSIGN	1BCE3	xMIN	1C520	xFC?C

1C559	xBIN	1CC1D	%12	1DBB0	BANGARRY
1C574	xDEC	1CC37	%13	1DC00	PUTLIST
1C58F	xHEX	1CC51	%14	1DD06	xV>
1C5AA	xOCT	1CC6B	%20	1DE66	x>V2
1C5C5	xSTWS	1CC85	%15	1DEC2	x>V3
1C5FE	xRCWS	1CCA4	%21	1E04A	xINDEP
1C619	xRCLF	1CCC3	%22	1E07E	xPMIN
1C637	RCLSYSF	1CCE2	%23	1E09E	xPMAX
1C64E	RCLUSERF	1CD01	%24	1EOBE	xAXES
1C67F	xSTOF	1CD20	%25	1EOE8	xCENTR
1C6A2	DOSTOALLF0	1CD3A	%16	1E126	xRES
1C6CF	STOALLFcont	1CD54	%17	1E150	x*H
1C6E3	DOSTOSYSF	1CD69	BINT_AfH	1E170	x*W
1C6F7	STOUSERF	1CD73	%26	1E190	xDRAW
1C731	STOSYSF	1CD8D	%27	1E1AB	xAUTO
1C783	x>LIST	1CDF2	%18	1E1C6	xDRAX
1C79E	xR>C	1CE07	%19	1E1E1	xSCALE
1C7CA	xRE	1CE28	xVTYPE	1E201	xPDIM
1C819	xIM	1CEE3	xEQ>	1E22B	xDEPND
1C85C	xSUB	1CF2E	SPLITEQ	1E25F	xERASE
1C8BB	XEQSUB\$	1CF42	drppargsym	1E27A	xPX>C
1C8EA	xREPL	1CF7B	xOBJ>	1E29A	xC>PX
1C95A	xLIST>	1CFD0	EXPR>	1E2BA	xGRAPH
1C973	XEQLIST>	1D009	x>ARRY	1E2D5	xLABEL
1C98E	xC>R	1D02C	XEQ>VECTOR	1E2F0	xPVIEW
1C9B8	xSIZE	1D054	XEQ>ARRAY	1E31A	xPIXON
1CA0D	DROP%1	1D054	XEQ>ARRY	1E344	xPIXOFF
1CA26	LEN\$>%	1D092	xARRY>	1E36E	xPIX?
1CA3A	LENCOMP>%	1D0AB	DOARRY>	1E398	xLINE
1CAB4	xPOS	1D0DF	xRDM	1E3C2	xTLINE
1CAD7	XEQPOS\$	1D186	xCON	1E3EC	xBOX
1CB0B	x>STR	1D2DC	xIDN	1E416	xBLANK
1CB26	xSTR>	1D392	xTRN	1E436	xPICT
1CB46	xNUM	1D407	xPUT	1E456	xGOR
1CB66	xCHR	1D5DF	xPUTI	1E4E4	xGXOR
1CB86	xTYPE	1D7C6	xGET	1E572	xLCD>
1CB90	XEQTYPE	1D8C7	xGETI	1E58D	x>LCD
1CC03	%11	1DABB	#1+ROT	1E5AD	x>GROB

1E5D2	xARC	1F27A	XEQINTEG	1FE48	xSUMY2
1E606	xTEXT	1F38B	SYMWHERE	1FE63	xSUMXY
1E621	xXRNG	1F3F3	xWHERE	1FE7E	xMAXSIGMA
1E641	xYRNG	1F439	XEQSYMWHERE	1FE99	xMEAN
1E661	xFUNCTION	1F43E	CKWHEREARGS	1FEB4	xMINSIGMA
1E681	xCONIC	1F500	xQUOTE	1FECF	xSDEV
1E6C1	xPARAMETRIC	1F5C5	xAPPLY	1FEEA	xTOT
1E6E1	xTRUTH	1F640	xFCNAPPLY	1FF05	xVAR
1E701	xSCATTER	1F9C4	x->Q	1FF20	xLR
1E721	xHISTOGRAM	1F9E9	x->QPI	1FF7A	xPREDV
1E741	xBAR	1FA59	xMATCHUP	1FF9A	xPREDY
1E761	xSAME	1FA8D	xMATCHDN	1FFBA	xPREDX
1E783	xAND	1FAEB	xFORMUNIT	1FFDA	xXCOL
1E809	xOR	1FB5D	xPREDIV	1FFFA	xYCOL
1E88F	xNOT	1FB87	xDUP	2001A	xUTPC
1E8F6	xXOR	1FBA2	xDUP2	2003A	xUTPN
1E972	x==	1FBBD	xSWAP	2005A	xUTPF
1EA9D	x#?	1FBD8	xDROP	2007A	xUTPT
1EBBE	x<	1FBF3	xDROP2	2009A	xSIGMACOL
1EC5D	x>	1FC0E	xROT	200C4	xSCLSIGMA
1ECFC	x<=?	1FC29	xOVER	200F3	xSIGMALINE
1ED9B	x>=?	1FC44	xDEPTH	2010E	xBINS
1EE38	xOLDPRT	1FC64	xDROPN	20133	xBARPLOT
1EE53	xPR1	1FC7F	xDUPN	20167	xHISTPLOT
1EE6E	xPRSTC	1FC9A	xPICK	2018C	xSCATRPLT
1EE89	xPRST	1FCB5	xROLL	201B1	xLINFIT
1EEA4	xCR	1FCD0	xROLLD	201D6	xLOGFIT
1EEBF	xPRVAR	1FCEB	xCLEAR	201FB	xEXPFIT
1EF43	xDELAY	1FDOB	xSTOSIGMA	20220	xPWRFIT
1EF63	xPRLCD	1FD2B	xCLSIGMA	2025E	xBESTFIT
1EFD2	xDER	1FD46	xRCLSIGMA	202CE	xSINV
1F0F5	XEQSYMDERSTEP	1FD61	xSIGMA+	2034D	xSNEG
1F113	XEQSYMDERCON	1FD8B	xSIGMA-	203CC	xSCONJ
1F133	xRCEQ	1FDA6	xNSIGMA	2044B	xSTO+
1F14E	xSTEQ	1FDC1	xCORR	20538	xSTO-
1F16E	xROOT	1FDDC	xCOV	2060C	xSTO/
1F201	XEQINTEGID	1FE12	xSUMY	20753	xSTO*
1F223	xINTEGRAL	1FE2D	xSUMX2	208F4	xINCR

209AA	xDECR	21B4E	DOAPWL	22EC3	xIF
20A15	xCOLCT	21B5A	XEQIOBACKUP	22EFA	xTHEN
20A49	xEXPAN	21C47	MEMSKIP	22F68	SYMCRUNCH1
20A7D	xRULES	21C6F	XEQSETLIB	22F86	SYMCRUNCH2
20A93	xISOL	21CBA	SETROMPART	22FB5	xELSE
20AB3	xQUAD	21E75	xXMIT	22FD5	xIFEND
20AD3	xSHOW	21E95	xSRECV	22FEB	xALG->
20B00	XEQSHOWLS	21EB5	xOPENIO	23033	xWHILE
20B20	xTAYLR	21ED5	xCLOSEIO	2305D	xREPEAT
20B40	xRCL	21EF0	xSEND	230C3	xDO
20B81	XEQRCL	21F24	xKGET	230ED	xUNTIL
20B9A	LISTRCL	21F62	xRECN	23103	xSTART
20CAD	PICTRCL	21F96	xRECV	231A0	xSTARTVAR
20CCD	xSTO	21FB6	xFINISH	2324C	xNEXT
20D65	xDEFINE	21FD1	xSERVER	23380	xSTEP
20EFE	xPURGE	21FEC	xCKSM	233DF	xIFERR
20F8A	XEQPURGEPICT	2200C	xBAUD	23472	xHALT
20FAA	xMEM	2202C	xPARITY	2349C	xSILENT'
20FD9	xORDER	2204C	xTRANSIO	234C1	xRPN->
20FF2	XEQORDER	2206C	xKERRM	235FE	x>>ABND
210FC	xCLUSR	22087	xBUFLEN	2361E	x<<
2115D	xTMENU	220A2	xSTIME	23639	x>>
21196	xMENU	220C2	xSBRK	23654	x'
211B4	ID_CST	220DD	xPKT	23679	xENDTIC
211E1	xRCLMENU	22352	%1200	23694	xWHILEEND
211FC	xPVARs	22367	%2400	236B9	xENDDO
2123A	xPGDIR	2237C	%4800	2371F	xERRTHEN
2125A	xARCHIVE	22391	%9600	23768	CKOATTNABORT
2133C	xRESTORE	224CA	xINPUT	2378D	xCASE
2137F	xMERGE	224F4	xASN	237A8	xTHENCASE
213D1	xFREE	22514	xSTOKEYS	23813	xDIR
2142D	xLIBS	22548	xDELKEYS	23813	xGROB
21448	xATTACH	22586	xRCLKEYS	23824	xPROMPT
2147C	xDETACH	225A4	ID_S	238A4	palparse
21660	SWAPDROPTRUE	225BE	x->TAG	23EED	ONE{ }N
21674	>BAK	225F5	USER\$>TAG	24COD	List
216D8	OB>BAKcode	22618	%>TAG	2512D	delimcase
21922	GetBVars	22633	xDTAG	2520A	'Rapndit

25223	DaDGNTc	29FD0	POP1%SPLITA	2A596	%%10
25322	4psh	29FDA	POP1%	2A5B0	%%>%
2534A	nultrior	2A002	POP2%	2A5C1	%%>%%
25446	tok->	2A188	PUSH%	2A5D2	SETDEG
25452	getmatchtok	2A23D	PUSH%LOOP	2A5F0	SETRAD
255BD	ngsizecase	2A2B4	%0	2A604	SETGRAD
255FB	need'case	2A2C9	%1	2A622	%D>R
25632	newBASE	2A2DE	%2	2A62C	PI/180
256E4	convertbase	2A2F3	%3	2A655	%R>D
25B0B	#+OVF	2A308	%4	2A673	%%>HMS
25C41	Extobcode	2A31D	%5	2A68C	%%HMS>
25CE1	GETPTRTRUE	2A332	%6	2A6A0	%%HMS+
26162	GetNextToken	2A347	%7	2A6C8	%%HMS-
265ED	realPACode	2A35C	%8	2A6DC	%%MAX
267D5	GPPushFTLp	2A371	%9	2A6F5	%%MAX
267DC	PushFTLp	2A386	%-1	2A70E	%%MIN
26A2D	?OKINALG	2A39B	%-2	2A727	%%0<
26FAE	GETPTRFALSE	2A3B0	%-3	2A738	%0<
27224	DNOTBAKcase	2A3C5	%-4	2A75A	%%0=
27234	DNOTLIBcase	2A3DA	%-5	2A76B	%0=
27244	NOTLISTcase	2A3EF	%-6	2A788	%%0>
27254	NOTSECOcase	2A404	%-7	2A799	%0>
27264	NOTROMPcase	2A419	%-8	2A7BB	%%0<>
27D00	check_pdata	2A42E	%-9	2A7CF	%0<>
28296	metatail	2A443	%PI	2A7E3	%%0>=
283C4	COLASkipcola	2A458	%%PI	2A7F7	%0>=
283D8	3SKIP	2A472	%%MAXREAL	2A80B	%%0<=
28558	#1-UNROT	2A487	%%-MAXREAL	2A81F	%%<
2856C	tok=casedrop	2A49C	%%MINREAL	2A871	%<
29A8D	nextsym'R	2A4B1	%%-MINREAL	2A87F	%%>
29BC2	subpdcdptch	2A4C6	%%0	2A88A	%>
29DCC	POSFLOWERR	2A4E0	%%1	2A895	%%>=
29DDC	NEGFLOWERR	2A4FA	%%2	2A8A0	%>=
29DEC	OVERFLOWERR	2A514	%%3	2A8AB	%%<=
29DFC	SETIVLERR	2A52E	%%4	2A8B6	%<=
29E0C	INFRESERR	2A548	%%5	2A8C1	%=
29E21	PACKSB	2A562	%%.1	2A8CC	%<>
29E46	PACK	2A57C	%%.5	2A8D7	%%SGN

2A8F0	%%ABS	2AC17	%%SINDEG	2AFC2	%RAN
2A900	%ABS	2AC27	%%SINRAD	2B044	%RANDOMIZE
2A910	%%CHS	2AC40	%COS	2B07B	DORANDOMIZE
2A920	%CHS	2AC57	%%COS	2B0C4	%FACT
2A930	%MANTISSA	2AC68	%%COSDEG	2B1FF	%%7
2A943	%%+	2AC78	%%COSRAD	2B2DC	%%12
2A94F	%%-	2AC91	%TAN	2B300	%%60
2A95B	%>%%-	2ACA8	%%TANRAD	2B3DD	%%.4
2A974	%+	2ACC1	%ASIN	2B45C	2%>%%
2A981	%-	2ACD8	%%ASINRAD	2B470	2%>%
2A99A	%%*	2ACF1	%ACOS	2B48E	%REC>%POL
2A9BC	%*	2AD08	%%ACOSRAD	2B498	%%R>P
2A9C9	%OF	2AD21	%ATAN	2B4BB	%POL>%REC
2A9E8	%%/	2AD38	%ANGLE	2B4C5	%%P>R
2A9FE	%/	2AD4F	%%ANGLE	2B4F2	%SPH>%REC
2AA0B	%T	2AD5B	%>%%ANGLE	2B529	RNDXY
2AA30	%CH	2AD6C	%%ANGLEDEG	2B53D	TRCXY
2AA5F	%%^	2AD7C	%%ANGLERAD	2B67D	aMODF
2AA70	%^	2AD95	%%SINH	2B770	aH>HMS
2AA81	%NROOT	2ADAE	%SINH	2B789	1/X15
2AA92	%%1/	2ADC7	%%COSH	2B7A7	RSUB1
2AA9E	%>%%1/	2ADDA	%COSH	2B7B0	RADD1
2AAAF	%1/	2ADED	%TANH	2B7CA	RADDF
2AAEA	%%SQRT	2AE00	%ASINH	2B7DC	ADDF
2AAF6	%>%%SQRT	2AE13	%ACOSH	2B91E	MULTF
2AB09	%SQRT	2AE26	%ATANH	2B977	DIVF
2AB1C	%%EXP	2AE39	%EXPONENT	2BA0F	SQRF
2AB2F	%EXP	2AE4C	%NFACT	2BBB5	DIV2
2AB42	%EXPM1	2AE62	%COMB	2BBE2	CLRFRC
2AB5B	%%LN	2AE75	%PERM	2BC4A	SPLITA
2AB6E	%LN	2AF27	%%H>HMS	2BCA0	SPLTAC
2AB81	%LOG	2AF4D	%FP	2BD32	Y<=X
2AB94	%%LNP1	2AF60	%IP	2BD76	TST15
2ABA7	%LNP1	2AF73	%CEIL	2BE53	XYEX
2ABBA	%ALOG	2AF86	%FLOOR	2BE61	STAB0
2ABDC	%MOD	2AF99	%%FLOOR	2BE6F	STAB2
2ABEF	%SIN	2AF99	%%INT	2BE7D	STCD0
2AC06	%%SIN	2AFAC	%INT	2BE8B	STCD2

2BE99	EXAB0	2CCBA	ColumnMAX	2E4DC	APNDCRLF
2BEA7	EXAB2	2CCD3	ColumnMIN	2E4F0	CRLF\$
2BEB5	RCAB0	2CCDF	ColumnTOT	2E5AB	SENDNAME
2BEC0	RCAB2	2CCEE	ColumnMEAN	2E6BE	InitIOEnv
2BECB	RCCD0	2CCFD	ColumnVAR	2E6EB	SENDLIST
2BED6	RCCD2	2CD09	ColumnTDEV	2E7EF	GETNAME
2BEE1	CCSB1	2D396	MAXRETRY	2E876	DOFINISH
2BEEC	RNDC[B]	2D3A0	LAMPKNO	2E8D1	DOPKT
2BFE3	GETAB1	2D3B1	LAMPACKET	2E99E	StdIOPAR
2BFFD	GETAB0	2D3C6	LAMRETRY	2E9CB	StoIOPAR
2C031	GETCD0	2D3D9	LAMERRMSG	2E9E6	SysSTO
2C04B	PUTAB0	2D3EE	LAMKP	2EA4F	GetIOPAR
2C1F3	STATSTO	2D3FB	LAMLNAME	2EA6A	Sys@
2C1FD	ID_SIGMADAT	2D40E	LAMOBJ	2EAE2	KERMOPEM
2C22F	STATCLST	2D41D	LAMOPOS	2EB37	DOOPENIO
2C270	STATRCL	2D42E	LAMEXCHP	2EB62	OpenIO
2C535	STATN	2D441	'LamKPSto	2EC11	%IP>#
2C558	STATSMAX	2D45A	LAMKLIST	2EC25	IOCheckReal
2C571	STATMEAN	2D46D	LAMKMODE	2EC34	SetIOPARerr
2C58A	STATSMIN	2D480	LAMKPTRN	2EC84	DOBAUD
2C5A3	STATSTDEV	2D493	LAMKRM	2ECCA	DOPARITY
2C5BC	STATTOT	2D4A2	LAMMaxR	2ED10	DOTRANSIO
2C5D5	STATVAR	2D517	EXCHINITPK	2ED4C	DOCKSM
2C675	STATCOL	2D564	Loop	2EDA6	DOKERRM
2C6B6	STATXCOL	2D58C	SENDEOT	2EDE1	DOBUFLEN
2C6CF	STATYCOL	2D5E1	SEND_PACKET	2EDF5	DOSTIME
2C6F2	STATGETXCOL	2D730	KDispRow2	2EE18	DOSBRK
2C706	STATGETYCOL	2D74E	KDispStatus2	2EE6F	DOXMIT
2C83C	STATCORR	2D816	DORECN	2EE97	DOSRECV
2C8E6	STATCOV	2D9A1	SetServMode	2EEC4	SendSetup
2C940	STATX	2D9B2	ClrServMode	2EFD7	TRPACKETFAIL
2C959	STATY	2D9F5	DOSERVER	2F211	LAMKML
2C972	STATXX	2DDC4	DropSysErr\$	2F383	IncrLAMPKNO
2C99A	STATYY	2E0A9	SENDNAK	2F39C	GetKermPkt#
2C9C2	STATXY	2E0C7	SENDERERROR	2F542	2DROPTTRUE
2CA0D	STATLR	2E0F4	SENDPKT	2F934	FalseFalse
2CADA	STATPREDY	2E108	BUILDKPACKET	2F989	RecvNextPkt
2CB4D	STATPREDX	2E31F	Push#FLoop	2FEA1	SENDAK

2FEB5	SENDNULLACK	32161	PRINT	35D35	MATIDN
2FEC9	KVISLF	32387	PRINTxNLF	35DEB	MATNEG
2FEDD	KVIS	323E9	PSubErr	35E2C	MATRND
2FFB4	GetStrLenStk	323F9	REMAP	35EA9	MATTRNC
2FFB7	GetStrLenC	324A6	AllowPr1cdC1	35F30	MATCONJ
2FFBA	GetStrLen	3251C	PopASavptr	35F8F	MATRE
3016B	KINVISLF	3251F	PopSavptr	35FA3	DUP%OCON
3037A	VERSTRING?	3252B	SetEcma94	35FEE	MATIM
30794	VERSTRING	325AA	ChkLowBat	36039	MATR>C
307E2	GETKP	32B08	ErrFixEIRU	360B6	MATC>R
30805	SUB\$1#	32B1A	FixEIRU	36115	MAT+
30914	ACK_INIT	32B74	PrintGrob	36278	MAT-
30B1D	CHOOSE_INIT	32CAF	ChkGrHook	362DC	MATFLOAT*
30BBE	ENCODE1PKT	32CB6	adrGraphPrtH	363DB	MATFLOAT/
30BD7	ENCODE	32FF9	SYMBNUMSOLVE	36444	MATSQ
30D31	DECODE	3303F	NUMSOLVE	3644E	MAT*
30E4E	PutSerialEck	3306C	7NULLLAM{}	365BB	MATRSD
30ED2	OUTUART	34301	Attn#	36705	MATDOT
31289	POPUART	34301	ATTN#	36791	MATCROSS
312DA	adr_uart_han	34D2B	1NULLLAM{}	368F4	MATRNRM
3133B	UARTBUFLEN	34D30	NULLLAM	3690D	MATCNRM
3136C	FLUSHRSBUF	353AB	FINDVAR	369E9	MATABS
31416	WaitTbz0	35491	apndvarlst	36A48	MATDET
31444	PUTSERIAL	3558E	PULLEL	36A99	MATINV
314E5	GETSERIAL	355B8	PULLREALEL	36AC3	MAT/
315C6	CLOSEUART	355C8	PULLCMPEL	3742D	!MAKEHXS
315F9	CloseUart	355D8	PULLLONGEL	3745E	SWAPROWS
31608	OpenUart?Clr	35602	PULLEREALEL	37508	SWAPCOLUMNS
3161E	OpenUartClr	35628	PUTEL	37B17	3NULLLAM{}
31854	docr	3566F	PUTREALEL	37B44	CKREF
31868	DOCR	356F3	PUTCMPEL	37BCB	ONE_DO_ARRAY
3187C	OpenIOPrt	357A8	MDIMS	37BE9	L%+
31EE2	DOPRLCD	35B47	SWITCHFLOATS	37C0C	L%-
31F4A	StdPRTPAR	35B88	SWITCH2FLOATS	37C2F	L%*
31F7D	StoPRTPAR	35C2C	DOARRAYPRG1	37C52	L%/
31FFD	DODELAY	35C63	DOARRAYPRG2	37C75	B%ABS
3205C	GetChkPRTPAR	35CAE	MATCON	37CD3	B%NEG
320B1	%80	35D08	DROP3PICK	37D19	F%>L%

37DB9	2NULLLAM{}	38C88	NoAppExitCnd!	38FFF	SetDA2bValid
37DF6	B%0=	38C98	AppError!	39018	SetDA3Valid
37E0F	MATREDIM	38CAB	AppError@	3902C	SetDA1Temp
37E0F	METREDIM	38CBE	NoAppError!	39045	SetDA2aTemp
3811F	MATTRN	38CD1	AppError?	39059	SetDA2bTemp
3858E	StartupProc	38CDF	SetAppError	39072	SetDA3Temp
385E8	InitSysUI	38CDF	SetBadPOLUI	39086	SetDA2Echo
3866F	SysMenuCheck	38CED	ClrAppError	39086	SetDA2aEcho
386A1	SysDisplay	38CFB	AppMode?	39086	?SetEditRoll
386D8	?FlashAlert	38D09	SetAppMode	390A4	MENoP&FixDA1
38728	SysErrorTrap	38D17	ClrAppMode	390B3	MENP&FixDA12
38908	DoWarning	38D25	NAppKeyOK?	390CC	ClrDA10K
38926	FlashWarning	38D33	SetNAppKeyOK	390E5	ClrDA2aOK
38985	ParOuterLoop	38D41	ClrNAppKeyOK	390FE	ClrDA2bOK
38994	POLSet&KeyUI	38D4F	DoStdKeys?	39117	ClrDA20K
389BC	POLSaveUI	38D5D	SetDoStdKeys	3912B	ClrDA30K
389CB	RclUI	38D6B	ClrDoStdKeys	39144	ClrDAsOK
38A11	RclHPUI	38D79	AppSuspOK?	3915D	SetDA2Valid
38A3E	LAMSavedUI	38D8A	SetAppSuspOK	39171	SetDAsValid
38A3E	SavedUILS	38D9B	ClrAppSuspOK	3918A	SetDA2NoCh
38A64	POLSetUI	38DAC	DA10K?	3919E	SetDA12NoCh
38AEB	POLKeyUI	38DE8	SetDA1BadT	391B2	SetDA23NoCh
38B09	1POLKeyUI	38DFC	DA2aOK?	391C6	SetDA13NoCh
38B45	POLKeyErr	38E38	SetDA2aBadT	391DA	SetDA12a3NoCh
38B45	POLErrorTrap	38E4C	DA2bOK?	391DA	SetDA12a3NCh
38B77	POLResUI&Err	38E88	SetDA2bBadT	391EE	SetDAsNoCh
38B90	POLRestoreUI	38E9C	DA20K?	391EE	SetDA123NoCh
38BD6	NormAppFlags	38EB5	DA30K?	39207	SetDA20KTemp
38BD6	InitPOLVars	38EF1	SetDA3BadT	3921B	SetDA12Temp
38C08	AppDisplay!	38F05	DAsOK?	3922F	SetDAsTemp
38C18	AppDisplay@	38F28	DA10K?NOTIT	39248	DAsBad?
38C28	NoAppDisplay!	38F41	DA2aOK?NOTIT	39275	DA1ValidF?
38C38	AppKeys!	38F5A	DA2bOK?NOTIT	39283	SetDA1ValidF
38C48	AppKeys@	38F73	DA30K?NOTIT	39291	ClrDA1ValidF
38C58	NoAppKeys!	38F8C	InitDispModes	3929F	DA2aValidF?
38C58	AppKeys0	38FB9	DA2aLess10K?	392AD	SetDA2aValidF
38C68	AppExitCond!	38FD2	SetDA1Valid	392BB	ClrDA2aValidF
38C78	AppExitCond@	38FEB	SetDA2aValid	392C9	DA2bValidF?

392D7	SetDA2bValidF	394CF	SetDA2bBad	39FB0	AbbrevStack?
392E5	ClrDA2bValidF	394DD	ClrDA2bBad	39FC1	SetAbbrevStk
392F3	DA3ValidF?	394EB	DA3Bad?	39FD2	ClrAbbrevStk
39301	SetDA3ValidF	394F9	SetDA3Bad	3A00D	DispEditLine
3930F	ClrDA3ValidF	39507	ClrDA3Bad	3A1CA	?DispMenu
3931D	DA1TempF	39515	DA1IsStatus?	3A1E8	DispMenu
3932B	SetDA1TempF	39515	DA1IsStat?	3A1FC	DispMenu.1
39339	ClrDA1TempF	39515	DispTime?	3A260	>MENU
39347	DA2aTempF?	39523	SetDA1IsStat	3A260	StdLabelDef
39355	SetDA2aTempF	39531	ClrDA1IsStat	3A297	Grob>Menu
39363	ClrDA2aTempF	3953F	DA2bIsEdL?	3A2B5	Str>Menu
39371	DA2bTempF?	3953F	DA2bEdit?	3A2C9	Seco>Menu
39371	DA2bTemp?	3954D	SetDA2bIsEdL	3A2DD	Id>Menu
3937F	SetDA2bTempF	3954D	SetDA2bEdit	3A328	MakeStdLabel
3938D	ClrDA2bTemp	3955B	ClrDA2bEdit	3A337	NullMenuLbl
3938D	ClrDA2bTempF	3955B	ClrDA2bIsEdL	3A38A	MakeBoxLabel
3939B	DA3TempF?	39569	NoRollDA2?	3A399	BoxLabelGrob
393A9	SetDA3TempF	3957A	SetNoRollDA2	3A3EC	MakeDirLabel
393B7	ClrDA3TempF	3958B	ClrNoRollDA2	3A3FB	DirLabelGrob
393C5	DA1NoCh?	3959C	?DispStatus	3A44E	MakeInvLabel
393D3	SetDA1NoCh	395BA	DispStatus	3A45D	InvLabelGrob
393E1	ClrDA1NoCh	39632	Blank&GROB!	3A4AB	MakeLabel
393EF	DA2aNoCh?	39673	AngleStatus	3A4CE	Disp5x7
393FD	SetDA2aNoCh	396C8	ComVecStatus	3A546	BlankDA1
3940B	ClrDA2NoCh	39748	UserFlagStat	3A55F	BlankDA2
39419	DA2bNoCh?	397BB	UserKeysStat	3A578	BlankDA12
39427	SetDA2bNoCh	3981B	AlgEntryStat	3A591	BlankDA2a
39435	ClrDA2bNoCh	39853	PrgmEntrStat	3A71C	DoNextRow
39443	DA3NoCh?	3988B	DispDir?Time	3A735	DoPrevRow
39451	SetDA3NoCh	398F4	DispDir?Tim1	3A9B8	'DROPFALSE
3945F	ClrDA3NoCh	39958	DispDir?Tim2	3A9CE	TurnOffKey
3946D	DA1Bad?	39971	PathStatus	3A9E7	SetSomeRow
3947B	SetDA1Bad	39AF1	DispTimeReq?	3AA0A	1A/LockA
39489	ClrDA1Bad	39B0A	DispStsBound	3ADED	DoPlotMenu
39497	DA2aBad?	39B2D	LineGrob	3B211	DoFirstRow
394A5	SetDA2aBad	39B85	?DispStack	3BDFA	EditMenu
394B3	ClrDA2aBad	39BF3	?RollUpDA2	3BE54	DoSolvrMenu
394C1	DA2bBad?	39F6F	LINESOFSTACK	3E2DD	<SkipKey

3E35F	>SkipKey	3FF86	SetNUsrKeyOK	40F86	InitMenu
3E3E1	<DelKey	3FF97	ClrNUsrKeyOK	41008	StartMenu
3E4CA	>DelKey	4019D	StdMenuKeyNS	410C6	SetThisRow
3E586	TogInsertKey	401D4	StdMenuKeyLS	41111	CheckMenuRow
3E5CD	IStackKey	40337	DoNameKeyRS	41175	LoadTouchTbl
3EC71	NullMenuKey	40350	DoNameKeyLRS	4139B	SaveLastMenu
3EC85	NoExitAction	40454	DoKeyOb	41422	>Review\$
3EC99	Box/StdLabel	40788	TakeOver	414BD	\$_:
3ECB2	Box/StdLbl:	40792	SystemLevel?	415C9	GetMenu%
3ECD0	FStd/BoxLbl:	407FB	MenuMaker	415F1	%100
3ECEE	FBox/StdLbl:	40828	MenuKey	41679	InitMenu%
3ED0C	Std/BoxLabel	4085A	Modifier	41741	InitTrack:
3ED25	BBox/StdLbl:	40882	CkSecoType	417F3	SetRebuild
3ED48	MBox/StdLbl:	408AA	ImmedEntry?	41848	MenuRow!
3ED6B	DirLabel:	40A6F	KeyOb!	4185B	MenuRow@
3EDA2	TogSysFlag	40A82	KeyOb@	4186E	LastMenuRow!
3EDF2	Do1st/1st+:	40A95	KeyOb0	41881	LastMenuRow@
3EE1A	Do1st/2nd+:	40AD9	Parse.1	418A4	MenuDef@
3EE47	Echo2Macros	40B2E	ParseFail	418D4	MenuRowAct!
3F78C	DUPExitAtLOOP	40B56	DispBadToken	418F4	LabelDef!
3F7EB	ExitAtLOOPDUP	40BC9	AtUserStack	41904	DoLabel
3FA57	Key>U/SKeyOb	40BDD	GetLastEdit	41914	MenuKeyNS!
3FA7A	?Key>UKeyOb	40C76	SaveLastEdit	41924	MenuKeyNS@
3FACF	ID_SKEY	40C94	CMDSTO	41934	DoMenuKeyNS
3FAE8	LAM_SKEY	40CE9	CacheStack	41944	MenuKeyLS!
3FB1A	Key>StdKeyOb	40D25	LockAlpha	41964	MenuKeyRS!
3FCAF	SetKeysNS	40D39	UnLockAlpha	41984	ReviewKey!
3FDDB	2DropBadKey	40D93	NoEdit?case	419C4	MenuExitAct!
3FDC7	DropBadKey	40DC0	DoMenuKey	419E4	LastMenuDef!
3FDD1	DoBadKey	40DD4	DoDelim	419F4	LastMenuDef@
3FDFE	'DoBadKey	40DF7	DoDelims	41A39	'idUserKeys
3FE12	'DoBadKeyT	40E3D	?ClrAlg	41A43	ID_UserKeys
3FE26	H/WKey>KeyOb	40E5B	?ClrAlgSetPr	41A5F	'idUserKeys.
3FE44	H/W>KeyCode	40E88	REPEATER	41A69	ID_UserKeys.
3FE7B	ModifierKey?	40EE7	SLOW	41A8D	UserKeys?
3FF1B	?CaseKeyDef	40F02	VERYSLOW	41AA1	Ck&AsnUKeys
3FF48	?CaseRomptr@	40F12	VERYVERYSLOW	41B28	XEQAsnKey
3FF75	NonUsrKeyOK?	40F22	XEQStoKey	41B3C	XEQDelKeys

41B69	Ck&ClrUKey	426F1	LastERow?	45C2F	RowElt#
41B8C	DelKey	4272D	TopERow?	45D1F	GetElt
41C02	XEQRclKeys	427AF	Cursor&Disp	46409	CopyRowsUp
41CA2	Ck&DecKeyLoc	42AE4	EchoChrKey	4651C	CopyColsLeft
41D92	CodePl>%rc.p	42BB6	Echo\$NoChr00	46625	CopyRowsDown
41E32	StoUserKeys	42BD4	Echo\$Key	4677E	CopyColsRght
41E78	StoUserKeypatch	42C3D	CkChr00	4744F	'IDX
41F13	ClrUserKeys	42CFB	InsertEcho	47467	STOAPPLDATA
41F2C	UserKeys!	42D32	EditLevel1	47984	APPprompt1!
41F3F	GetUserKeys	42D46	ViewLevel1	479A7	APPprompt2
41F52	PgUserKeys	42D82	CharEdit	48FF9	SORTASLOW
41F65	WaitForKey	42DC8	ObEdit	491D5	AUTOSCALE
4203C	GetKeyOb	42E27	Roll&Do:	494B4	%.1
42078	?BlinkCursor	42E86	Rcl&Do:	495AA	%.15
420A0	GETKEY*	42EC7	AdjEdModes	49709	PromptIdUtil
420F5	ATTNxcp	42F44	InputLine	49AD3	PointDerivUt
42113	ALARMxcp	430CF	DispILPrompt	49BA5	FcnUtilEnd
42131	SLEEPxcp	43179	InputLEnter	49BD2	RootUtil
42145	UARTxcp	43200	InputLAttn	49C54	CkEQUtil
42159	GETKEY	4325A	SetCursor	49CD6	ROLL{}
42249	UART?	4334F	DRPExitAtLOOP	49F06	PointMoveCur
42262	ATTN?	44277	InitEd&Modes	4A055	DISPCOORD2
4226A	addrATTNFLG	4428B	InitEdLine	4A0AA	GetEqN
4227F	TIMEOUT?	44394	InitEdModes	4A95A	ICMPDRPRTDRP
42284	adrTIMEOUTCLK	443CB	EditString	4A9AF	CHECKPVARs
422A1	ALARM?	444A5	CURSOR_END?	4AAEA	MAKEPVARs
423BB	settimeout	444EE	Char>Edit	4AB1C	ID_X
423D3	clrtimeout	44683	EDITLINE\$	4AB2A	PvarsC%0
42402	KEYINBUFFER?	44730	EDITPARTS	4AB59	ID_Y
4243E	?ATTNQUIT	4488A	NoEditLine?	4ADB0	GETSCALE
4243E	?ATTN_QUIT	4489E	ROWNUM	4AE3C	PUTSCALE
4245C	NoAttn?Semi	448C1	?TogU/LCase	4AF63	GETINDEP
42475	DoCAAlarmKey	44C31	DoNewMatrix	4AF77	PUTINDEP
4248E	CtlAlarm!	44F42	BindMatVars	4AF8B	PUTINDEPLIST
424A1	CtlAlarm@	44FE7	DoOldMatrix	4AFDB	GETRES
424DA	DCursor	45023	InitOldMat	4B012	PUTRES
4256B	LCursor	45676	Blank\$	4B062	GETPTYPE
42660	UCursor	45AE0	GetMat/Vec	4B076	PUTPTYPE

4B0DA	GETPMIN&MAX	4E360	MENUOFF?	50B08	LINEOFF
4B10C	GETXMIN	4E37E	LINECHANGE	50B17	LINEON
4B120	GETYMIN	4E442	CURRENTMARK?	50D78	LASTPT?
4B139	GETXMAX	4E46A	EQCURSOR?	50DA5	PlotOneMore?
4B14D	GETYMAX	4E497	PREMARKON	50E59	!#1+IF<dim-1
4B166	PUTXMIN	4E4B0	NEWMARK	50EA5	!#1-IF>0
4B189	PUTYMIN	4E582	DRAWBOX#	50F60	dropDROPf
4B1AC	PUTXMAX	4E6EF	DispCoord1	510AD	INDEPVAR
4B1CF	PUTYMAX	4E776	Z-BOX	510D5	RECORDX&YC%
4B323	MAKEPICT#	4ECAD	CROSSMARKON	51125	CLEARMENU
4B364	GETPARAM	4F052	WINDOW#	51148	CKPICT
4B553	VSCALE	4F0AC	DOPX>C	51166	CHECKPICT
4B5AD	HSCALE	4F179	DOC>PX	511E3	CHECKHEIGHT
4B60C	DOERASE	4F1D8	SWAPTRUE	5127E	'IDPAR
4B6D9	PLOTERR	4F3D1	2HXS>#	5129C	'IDFUNCTION
4B710	RESETDEPTH	4F408	C%>#	512B0	'IDCONIC
4B765	PLOTPREP	4F78C	GROB+#	512C4	'IDPOLAR
4B7D8	GetRes	4F7E6	CKGROBFITS	512D8	'IDPARAMETER
4BFAE	NEXTSTEP	50262	%1+	512EC	'IDTRUTH
4C09B	NEWINDEP	50276	%1-	51300	'IDSCATTER
4C639	drax	503D4	DOLCD>	51314	'IDHISTOGRAM
4CE4C	EXITFCNsto	50438	DO>LCD	51328	'IDBAR
4CE6F	DOGRAPHIC	5046A	DOCLLCD	5133C	PtoR
4CEE7	GraphicExit	5053C	CROSSGROB	514AF	GETRHS
4CF05	GDISPCENTER	5055A	MARKGROB	514DC	1REV
4CF41	SETLOOPENV	50578	GROBDIM	514EB	%2PI
4CF68	ExitFcn	505B2	NULLPAINT	51532	2HXSLIST?
4D11E	DROPDEADTRUE	505C6	GETXPOS	515A0	TOPROW
4D132	SCROLLUP	505E4	getxpos	515B4	BOTROW
4D150	SCROLLLEFT	5068D	GETYPOS	515FA	LEFTCOL
4D16E	SCROLLEDOWN	506AB	getypos	5160E	RIGHTCOL
4D18C	SCROLLRIGHT	5072B	TOGGLELINE#3	5162C	WINDOWTOP?
4DA0D	CROSS_HAIRS	50758	DRAWLINE#3	51645	WINDOWBOT?
4DA76	CROSS_OFF	50A3B	UNROT2DROP%0	5165E	WINDOWLEFT?
4E266	CHECKMENU	50ACC	LINEOFF3	51677	WINDOWRIGHT?
4E2AC	MENUOFF	50ADB	TOGLINE3	51690	JUMPTOP
4E2CF	TURNMENUOFF	50AEA	LINEON3	516AE	JUMPBOT
4E347	TURNMENUON	50AF9	TOGLINE	516E5	JUMPLEFT

51703	JUMPRIGHT	51C84	SWAP2C%>%	52571	C%COS
51735	REPEATERCH	51C9D	C%%C+R	525B7	C%TAN
5179E	DUPGROBDIM	51CB1	C%%R+C	5262F	C%SINH
517F3	EQUALcasedrop	51CD4	C%R-C	52648	C%COSH
517FE	EQUALcasedro	51CE8	C%C-R	5265C	C%TANH
5182F	ISTOP-INDEX	51CFC	C%C-C	52675	C%ATAN
51843	SWAP#1+SWAP	51D10	C%%C-C	5267F	C%i
51857	SWAP#1-SWAP	51D24	C%%R-C	526AE	C%-i
5187F	GBUFFGROBDIM	51D38	C%%C-R	527EB	C%ATANH
51893	ORDERXY#	51D4C	C%C*R	52804	C%ASIN
518CA	ORDERXY%	51D60	C%R*C	5281D	C%ASINH
5193B	C%%1	51DAB	C%%C*R	52836	C%ACOSH
5196A	C%-1	51DBF	C%%R*C	52863	C%ACOS
5198F	DROP%0	51DE2	C%%C*C	52AB7	POPC%
519A3	C>Re%	51E19	C%R/C	52ADB	PUSHC%
519B7	C>Im%	51E64	C%C/R	52B57	POPC%%
519CB	C%>%	51EC8	C%C/C	52B95	PUSHC%%
519DF	C%>%%SWAP	51EFA	C%1/	52C36	CALCINTEG
519F8	C%%>C%	51F13	C%%C/C	52D26	4NULLLAM{}
51A07	%>C%	51F3B	C%%R/C	53725	SetUserFlag
51A37	Re>C%	51F7C	C%%C/R	53731	SetSysFlag
51A4A	C%*i	52062	C%ABS	53755	ClrUserFlag
51A5F	C/i	52080	C%%ABS	53761	ClrSysFlag
51A71	2%>%%SQR	52099	C%ARG	53778	TestUserFlag
51A94	%SQR	520B2	2DUP%%R	53784	TestSysFlag
51AB7	%MAXIMIZE	520CB	C%SGN	5380E	COERCEFLAG
51B2A	C%%0=	52107	C%SQRT	53860	HISTON?
51B43	C%0=	52193	C%EXP	538C0	UNDO_ON?
51B70	C%CHS	521E3	C%LN	538CE	UNDO_ON
51B91	C%%CHS	5226F	SYMUVAL	538DC	UNDO_OFF
51BB2	C%CONJ	522BF	C%LOG	538F8	NOBLINK
51BC1	C%%CONJ	52305	C%ALOG	53968	AlgEntry?
51BD0	C%C+R	52342	C%R^C	53976	SetAlgEntry
51BE4	%+SWAP	52360	C%C^R	53984	ClrAlgEntry
51BF8	C%R+C	52374	C%C^C	539F9	SetISysFlag
51C16	C%C+C	524AF	C%0	53A05	D0=ALoop
51C3E	C%%C+C	524F7	C%1	53A20	REPLACE_MODE
51C6B	SWAP2C%>%	52530	C%SIN	53A2E	INSERT_MODE

53A3C	INSERT?	53F05	bit/	54977	SYMDERSTEP
53A4A	EditLExists?	53F77	POP2HXS	5499F	ReplOABND
53A90	ClrNewEditL	53F8D	POP2HXS	549DB	lam'dvar
53AE4	NoIgnoreAlm	54021	getwordsize	549EC	MetaConcase
53B31	setflag	54039	WORDSIZE	54A9C	DROP%1ABND
53B61	NumbMode	54050	BASE	54AEO	SYMRE
53B88	CLRNUM	54061	HXS>\$	54B1E	DROP%OABND
53B9C	SETNUM	5407A	BASECHAR	54C4F	RDROPCOLATRUE
53BB0	NOTNUM?	540BB	hxs>\$	54C63	nmetasyms
53BC9	DEG?	5422C	PUSHhxs	54CB3	SWAPDROPLLOOP
53BDD	RAD?	54266	GPPushA	54CEF	Cr
53C0A	NOTCONST?	5429F	bit%#/	54CEF	SYMBN:
53C23	PRSOL?	542BD	bit%#/	54DBC	ckseval1:
53C37	DOHEX	542D1	bit%#*	54E2A	ckeval1eq1
53C43	DOBIN	542EA	bit%#*	54EB9	SYMIM
53C4F	DOOCT	542FE	bit%#-	54ED2	SYMNOT
53C5B	DODEC	5431C	bit%#-	54EEB	SYMNEG
53C96	XEQSTWS	54330	bit%#+	54F04	SYMABS
53CAA	dostws	54349	bit%#+	54F1D	SYMCONJ
53CF0	XEQRCWS	5435D	#>%	54F36	SYMINV
53D04	bitAND	5435D	HXS>%	54F4F	SYMARG
53D15	bitOR	543F9	%>#	54F68	SYMSIGN
53D26	bitXOR	544D9	HXS==HXS	54F81	SYMSQRT
53D4E	bitNOT	544EC	HXS#HXS	54F9A	SYMSQ
53D5E	bitSL	54500	HXS>HXS	54FB3	SYMSIN
53D6E	bitSLB	5452C	HXS>=HXS	54FCC	SYMCOS
53D81	bitSR	5453F	HXS<=HXS	54FE5	SYMTAN
53D91	bitSRB	54552	HXS<HXS	54FFE	SYMSINH
53DA4	bitRR	54564	SYMIFTE	55017	SYMCOSH
53DE1	bitRRB	545A0	cknumdsptch1	55030	SYMTANH
53E0C	bitRL	54609	MetaIFTE	55049	SYMASIN
53E3B	bitRLB	54653	NumIFTE	55062	SYMACOS
53E65	bitASR	547B5	SYMBWHERE	5507B	SYMATAN
53EA0	bit+	547E2	WHERE1	55094	SYMASINH
53EB0	bit-	5483C	COLAkeep1st	550AD	SYMACOSH
53EC3	bitNEG	54887	WHEREIN	550C6	SYMATANH
53ED3	bit*	548AA	revpull&psh	550DF	SYMEXP
53EE4	MPY	54954	SYMDER	550F8	SYMLN

55111	SYMLOG	55607	repl%-1	55C56	%SYM%OF
5512A	SYMALOG	55657	cknum2:	55C6F	SYM%OF
55143	SYMLNP1	5573D	COLAhexFCN	55C88	SYM%%CH
5515C	SYMEXPM	5575B	cknseval:	55CA1	%SYM%CH
55175	SYMFACT	5576F	cksneval2:	55CBA	SYM%CH
5518E	SYMIP	557EC	cksseval2:	55CD3	SYM%%T
551A7	SYMFP	558BE	?spliteq	55CEC	%SYM%T
551C0	SYMFLOOR	558DC	sscknum2	55D05	SYM%T
551D9	SYMCEIL	558F5	sncknum2	55D1E	SYM%COMB
551F2	SYMEXPONENT	5590E	nscknum2	55D37	%SYMCOMB
5520B	SYMMANT	5599A	SYM%AND	55D50	SYMCOMB
55224	SYMD>R	559B3	%SYMAND	55D69	SYM%PERM
5523D	SYMR>D	559CC	SYMAND	55D82	%SYMPERM
55256	SYMUBASE	559E5	SYM%OR	55D9B	SYMPERM
55288	1GETLAMSWP1+	559FE	%SYMOR	55DB4	SYM%RND
5529C	MetaSINH	55A17	SYMOR	55DCD	SYMRND
552B0	MetaCOSH	55A30	SYM%XOR	55DE6	RNDSYM
552C4	MetaTANH	55A49	%SYM XOR	55DFF	SYM%TRNC
552D8	MetaEXP	55A62	SYM XOR	55E18	TRCNYM
552EC	MetaALOG	55A7B	SYMFLOAT==	55E31	SYMTRCN
55300	MetaEXPM	55A94	FLOATSYM==	55E4A	SYM%MAX
55314	?addinver:	55AAD	SYM==	55E63	%SYM MAX
5533C	MetaSIN	55AC6	SYMFLOAT<>	55E7C	SYM MAX
55378	MetaCOS	55ADF	FLOATSYM<>	55E95	SYM%MIN
553A5	MetaTAN	55AF8	SYM<>	55EAE	%SYM MIN
553D2	MetaNEG	55B11	SYM%<	55EC7	SYM MIN
553EB	MetaINV	55B2A	%SYM<	55EE0	SYM^0
5540E	?addrevert	55B43	SYM<	55EF9	0^SYM
5542C	MetaRE	55B5C	SYM%>	55F12	SYM^
55477	replfunc	55B75	%SYM>	55F2B	SYM+0
55495	MetaIM	55B8E	SYM>	55F44	0+SYM
554B3	repl%1	55BA7	SYM%<=	55F5D	SYM+
5551C	Repl0	55BC0	%SYM<=	55F76	SYM-0
55535	Repl1	55BD9	SYM<=	55F85	0-SYM
5554E	Repl-1	55BF2	SYM%>=	55F8F	SYM-
55567	MetaCONJ	55C0B	%SYM>=	55FC1	SYM*0
555B2	MetaABS	55C24	SYM>=	55FDA	0*SYM
555E9	MetaSQ	55C3D	SYM%%OF	55FF3	SYM*

5600C	SYM/0	56949	SYMSUM	58CEE	eval
56025	0/SYM	56A06	SYM%SUM	58D75	SYMSHOW
5603E	SYM/	56A4C	%SYMSUM	5910B	SHOWLS
56057	SYM%MOD	56AC9	%SUM	5918A	evalTRUE
56070	%SYMMOD	56AFB	4DROP%0	591AD	SYMQUAD
56089	SYMMOD	57293	SYMISOL	595DD	SYMTAYLR
560A2	SYM%XROOT	57405	2DROP2dropf	596D3	MetaRCOLCT
560BB	%SYMXRROOT	57419	DROP2dropf	5971D	MetaDNEG
560D4	SYMXRROOT	57432	addtpsh	5976B	MetaDINV
560ED	xssgneral	5768A	0bInMeta?	597B5	Meta*1
560ED	xssgeneral	57A0C	SYMEXPAN	5983B	Meta^1
56101	xnsgeneral	57A48	expan	59885	Meta1/
5611F	xsgeneral	57AA2	expan1	5990F	Meta+1-1
5613D	?addsimir	57AB6	expansq	5A01D	SWAPcompSWAP
56160	Meta+	57B01	?expanapp	5A036	uncrunch
56174	Meta-	57B4C	?expanneginv	5A310	symbn
56183	2Repl0	57B63	?expan^	5AAC7	SYMINTEG
561BA	Meta*	57C71	expansum^	5AC86	dvars>
561D8	2Repl-1	57CF8	NXTPOT%	5ACC7	intg
56214	Meta/	57D90	SYMCOLCT	5ACD6	M1st+?Drp
56250	Metamod	57DA4	colct	5AD08	dvars?
562BE	dropaddoper	57E08	colrev	5AD6C	linear!
562FA	Con^Meta	57F4B	swappargunrot	5AD80	linear?
56309	MetaUnCalc	584B2	MEQU?	5AD9E	linear
56331	Con+Meta	58511	MetaLess?	5AFAB	intg1
56359	Con-Meta	58525	MetaMore?	5B09B	intg1fail
56390	Con*Meta	585A7	BodyMore?	5B0CD	intgconst
563DB	Con/Meta	58715	NoIdsInMeta?	5B0FA	intglok
5642B	ConmodMeta	587AA	colfac	5B131	intglinear
5643A	DropRepl0	58A20	colunfac	5B140	intgaddlin
5645D	Meta^Con	58A61	colinv1	5B659	forcemul?aga
56543	Meta+Con	58A93	colinv2	5B717	forcemul
56566	Meta-Con	58AAC	colinv3	5B861	top&pshtop&
56589	Meta*Con	58ADE	M-1potcase	5BBE5	metaneg
565CF	Meta/Con	58C02	count+	5BC03	metaneglft
5660B	MetamodCon	58C0E	count*	5BC21	metainvlft
5662E	NUMINTEG	58CDA	coleval	5BC3F	metainv
5670F	>dvars	58CE4	parameval	5BC5D	meta+

5BC67	addt-	5CC12	2pull2DROP	5E401	psh1top&
5BC8A	meta-	5CCEE	larg&	5E415	top&
5BC94	addt+	5CD16	addt*	5E490	2top&
5BCB7	meta*	5CD2A	addtNEG	5E4A9	#1-SWAP
5BCC1	repl/	5CD3E	addtINV	5E4A9	pull
5BCE4	meta/	5CD52	evalcase:	5E4BD	pullrev
5BCEE	repl*	5CD7A	revalcase:	5E4D1	pshtop&
5BD3E	drpmeta+	5CDF2	Meta<-Dall	5E4EA	pullpsh1&
5BD57	drpmeta-	5CE15	Meta<-Aall	5E503	pullrev1&
5BD70	drpmeta*	5CE4C	MetaA->all	5E51C	addt:
5BD89	drpmeta/	5CE83	Meta->)all	5E530	addt2:
5BE56	MetaMulInv	5CEBA	Meta(<-all	5E549	psh1&rev:
5BE81	Meta<-->	5CEF1	MetaD->all	5E562	psh1&rev2:
5BECE	Meta<-A	5CF23	MetaT->all	5E585	INNERtop&
5BF53	MetaA->	5CF5A	Meta<-Tall	5E59E	repl:
5BFD8	MetaD->	5CF91	Meta->()all	5E5B7	@OBNOT
5C0B9	Meta<-D	5CFC3	Meta->()C%all	5E5EE	SAFE@NOT
5C102	Meta<-D!	5CFF5	Meta<-Mall	5E602	@LAMNOT
5C137	Meta->()	5D009	MetaM->all	5E616	ROMPTR@NOT
5C204	Meta1/()	5D0C2	forceadd	5E652	symcomp
5C261	Meta-()	5D6FA	pshpullpsh1&	5E661	ONESYMBN
5C2CE	MetaE^	5DD65	2psh1&rev:	5E67A	pshzer
5C31B	MetaE()	5DE41	COLATRUE	5E68E	pargop
5C348	MetaL*	5DE55	RDROPFALSE	5E6BB	pullpshm1
5C375	MetaL()	5DE7D	reversym	5E6F2	parg&
5C3C2	Meta<-M	5E067	SINNER	5E706	psh1&
5C4CF	MetaM->	5E085	CKSYMBN	5E7A5	psh1&rev
5C53C	MetaAF	5E0A3	PSYMBN	5E843	overev&
5C589	Meta(<-	5E0DA	P{ }N	5E857	rotswap
5C5D6	Meta->)	5E111	P: :N	5E870	4rollunrot
5C623	Meta(())	5E2F8	2SINNERtop&	5E8DE	argum
5C670	Meta->TRG	5E30C	2SINNER	5E984	nonopcase
5C68D	MetaT->	5E32A	SINNERMETA	5E9A7	infarg?
5C6D9	Meta<-T	5E35C	dup	5EA09	CKINFARGS
5C73D	Meta->()C%	5E370	NDUPN	5EA9F	pshzerpsharg
5C845	Meta->DEF	5E3AC	psh&	5EAC2	larg
5C91D	MetaTRG*	5E3C0	over&	5EAF4	SWAPDROP#1-
5CBF9	drppargtop&	5E3E8	pshm1	5EAF4	pulldrop

5EB1C	psh	5F208	C%2=case	60F7E	XYZW>
5EB58	rot	5F23A	num-1=case	60F83	4DropLoop
5EBC6	4roll	5F267	%-1=case	60F9B	XY>Y
5EBDB	unrot	5F285	C%-1=case	60F9B	SWAPDROP
5EBEA	4unroll	5F2A3	EXPLODE	60FAC	3UNROLL
5EBFC	N+1roll	5F2EE	IMPLODE	60FAC	UNROT
5ED45	5roll	5F384	DROPSYM	60FAC	XYZ>ZXY
5ED5A	5unroll	5F5E4	4DROPFALSE	60FBB	XYZW>YZWX
5ED6C	N+1unroll	5F657	3DROPTRUE	60FBB	4ROLL
5EDFC	num1stcase	5F6B1	5DROPFALSE	60FBB	FOURROLL
5EE10	M-1stcasechs	5F926	splitup	60FD8	5ROLL
5EEDB	REALNEGcase	5F96E	splitdown	60FD8	FIVEROLL
5EF15	AEQ1stcase	5F996	tailpsh	61002	6ROLL
5EF2E	M1st+case	5FA45	pulldroppull	61002	SIXROLL
5EF41	M1st-case	5FA63	revpulldrop	6103C	EIGHTROLL
5EF54	M1st*case	5FB49	NOTcaseFALSE	6103C	8ROLL
5EF67	M1st/case	5FB76	ROT#1+UNROT	6106B	SEVENROLL
5EF7A	M1stNEGcase	5FBE6	pick1#0=case	6106B	7ROLL
5EF8D	M1stINVcase	5FC24	pick1	61099	DUP4UNROLL
5EFA0	M1st^case	5FC24	DUP#2+PICK	6109E	FOURUNROLL
5EFB3	M1stSQcase	5FC38	%1pshm1	6109E	XYZW>WXYZ
5EFC6	M1stFNCcase	60EBD	RSWAP	6109E	4UNROLL
5EFD9	MEQ1stcase	60EE7	XYZ>YXZ	610C4	5UNROLL
5EFF9	MEQopscase	60EE7	ROTSWAP	610C4	FIVEUNROLL
5F048	AEQopscase	60FOE	XYZ>ZY	610FA	6UNROLL
5F061	Mid1stcase	60FOE	ROTDROPSWAP	610FA	SIXUNROLL
5F090	doskip	60FOE	UNROTSWAPDROP	6112A	XYZ>Z
5F09D	docola	60F21	XYZ>YZ	6112A	UNROT2DROP
5F0AA	idntcase	60F21	ROTDROP	6112A	ROTRROT2DROP
5F0CD	idntlamcase	60F33	XYZ>ZYX	6113C	4UNROLL3DROP
5F0FA	num0=case	60F33	UNROTSWAP	6113C	XYZW>W
5F127	%0=case	60F33	SWAPROT	6114E	2DROP
5F13B	C%0=case	60F4B	3DROP	61160	3DROP
5F154	num1=case	60F4B	XYZ>	61172	#-PICK
5F181	%1=case	60F54	7DROP	61184	#+PICK
5F19F	C%1=case	60F66	6DROP	6119E	DUP#1+PICK
5F1BD	num2=case	60F72	5DROP	611A3	#1+PICK
5F1EA	%2=case	60F7E	4DROP	611BE	#2+PICK

611D2	#3+PICK	6152C	18GETLAM	618F7	casedrop
611E1	#4+PICK	6153C	19GETLAM	61910	NOTcase2drop
611F9	2DUPSWAP	6154C	20GETLAM	6191F	case2drop
611F9	DUP3PICK	6155C	21GETLAM	61933	EQcase
611FE	3PICK	615E0	1PUTLAM	6194B	caseDROP
6121C	4PICK	615F0	2PUTLAM	61960	NOTcaseDROP
6123A	5PICK	61600	3PUTLAM	61970	case2DROP
6125E	6PICK	61610	DUP4PUTLAM	61984	NOTcase2DROP
61282	7PICK	61615	4PUTLAM	61993	case
612A9	8PICK	61625	5PUTLAM	619AD	NOTcase
612CC	#-ROLL	61635	6PUTLAM	619BC	IT
612DE	#+ROLL	61645	7PUTLAM	619CB	GOTO
612F3	#1+ROLL	61655	8PUTLAM	619E0	?GOTO
61305	get1	61665	9PUTLAM	619F3	NOT?GOTO
61318	#2+ROLL	61675	10PUTLAM	61A02	popflag
6132C	#-UNROLL	61685	11PUTLAM	61A18	#0=?SEMI
6133E	#+UNROLL	61695	12PUTLAM	61A2C	NOT?SEMI
61353	#1+UNROLL	616A5	13PUTLAM	61A3B	?SEMI
61365	#2+UNROLL	616B5	14PUTLAM	61A47	SEMILOOP
61380	DUPUNROT	616C5	15PUTLAM	61A58	ITE_DROP
61380	SWAPOVER	616D5	16PUTLAM	61A6D	COLA_EVAL
613B6	1GETLAM	616E5	17PUTLAM	61A8E	COLARPITE
613E7	2GETLAM	616F5	18PUTLAM	61AD8	ITE
6140E	3GETLAM	61705	19PUTLAM	61AE9	2'RCOLARPITE
61438	4GETLAM	61715	20PUTLAM	61B45	2@REVAL
6145C	5GETLAM	61725	21PUTLAM	61B55	3@REVAL
6146C	6GETLAM	61735	22PUTLAM	61B72	NOT?DROP
6147C	7GETLAM	61745	DUPTENV	61B89	ticR
6148C	8GETLAM	61745	DUPTMEVN	61C1C	EXPAND
6149C	9GETLAM	617D8	GETLAMPAIR	61CE9	CACHE
614AC	10GETLAM	6186C	#=case	61D3A	SAVELAM
614BC	11GETLAM	6187C	OVER#=case	61D41	SAVESTACK
614CC	12GETLAM	61891	DUP#0=case	61EA7	DUMP
614DC	13GETLAM	61896	#0=case	61F8F	undo
614EC	14GETLAM	618A8	DUP#0=csedrp	61FA9	DUPROM-WORD?
614FC	15GETLAM	618BA	EQcasedrop	61FB6	ROM-WORD?
6150C	16GETLAM	618D3	#=casedrop	61FCF	Rom-Word?
6151C	17GETLAM	618E8	NOTcasedrop	62001	2SWAP

62020	DUPTYPECHAR?	62193	DTYPEARRY?	624C6	#MAX
62025	TYPECHAR?	62193	DUPTYPEARRY?	624FB	#-#2/
62035	DUPTYPEIDNT?	62198	TYPEARRAY?	62535	DROPZERO
6203A	TYPEIDNT?	62198	TYPEARRY?	6254E	2DROP00
6204A	DUPTYPEEXT?	621A8	DUPTYPEROMP?	6256A	#3+
6204F	TYPEEXT?	621AD	TYPEROMP?	6257A	#4+
62073	GPOverWrT/FL	621BD	DUPTYPERRP?	6258A	#5+
62076	GPOverWrTLp	621C2	TYPERRP?	6259A	#6+
6207D	OverWrF/TLp	621D2	DUPTYPESYMB?	625AA	#7+
62080	OverWrTLoop	621D7	TYPESYMB?	625BA	#8+
62096	GPOverWrFLp	621E7	DUPTYPECOL?	625CA	#9+
6209D	OverWrT/FLp	621E7	DTYPECOL?	625DA	#10+
620A0	OverWrFLoop	621EC	TYPECOL?	625EA	#12+
620B6	GPPushT/FLp	621FC	DUPTYPEGROB?	625FA	#3-
620B9	GPPushTLoop	62201	TYPEGROB?	6260A	#4-
620C0	PushF/TLoop	62211	DTYPELIST?	6261A	#5-
620C3	PushTLoop	62211	DUPTYPELIST?	6262A	#6-
620D2	GPPushFLoop	62216	TYPELIST?	6264E	#10*
620D9	PushT/F	62226	DUPTYPETAG?	62674	#8*
620D9	PushT/FLoop	6222B	TYPETAGGED?	62691	#6*
620DC	PushFLoop	6223B	TYPERARRY?	626AE	5skipcola
620EB	OVER#=#	62256	TYPECARRY?	626DC	3skipcola
62103	DROPTRUE	62266	DUP#0=	626E5	2skipcola
6210C	DROPFALSE	62289	#3=	626EE	skipcola
62115	DUPTYPELAM?	6229A	#2=	626F7	DUP#2+
6211A	TYPELAM?	622A7	#1=	6270C	DROPSWAP
6212A	DUPTYPEBINT?	622B6	#1<>	62726	XYZ>Y
6212F	TYPEBINT?	622C5	DUP#1=	62726	DROPSWAPDROP
6213F	DUPTYPEHSTR?	622D4	DUP#0<>	62726	ROT2DROP
62144	TYPEHSTR?	622E5	!insert\$	62747	SWAPDUP
62154	DUPTYPECSTR?	622EF	SWAP&\$	62775	ROTDUP
62154	DTYPECSTR?	62312	!!append\$?	62794	SWAP#-
62159	TYPECSTR?	62376	!append\$	627A7	DROPDUP
62169	DUPTYPEREAL?	62394	!!insert\$	627BB	DUPLen\$
62169	DTYPEREAL?	623A0	!!append\$	627D5	#+DUP
6216E	TYPEREAL?	62474	'RSaVEWORD	627EB	Push2#aLoop
6217E	DUPTYPECMP?	62474	'RSaveRomWrd	627F8	#-DUP
62183	TYPECMP?	624BA	#MIN	62809	#1+DUP

6281A	#1-DUP	62C05	DUP@	62EDF	3PICKSWAP
62830	SWAPDROPDUP	62C19	DUPROMPTR@	62EF3	4PICKSWAP
6284B	SWAPDROPSWAP	62C2D	#=ITE	62F07	1GETSWAP
6284B	XYZ>ZX	62C41	INNERDUP	62F1B	?SWAP
6284B	UNROTDROP	62C55	NOTAND	62F2F	!append\$SWAP
62864	4ROLLDROP	62C69	TOTEMPSWAP	62F43	NOT?SWAPDROP
62880	5ROLLDROP	62C7D	ROT2DUP	62F5C	?SWAPDROP
6289B	2DUP#<	62C91	ROTAND	62F75	#1+NDROP
628B5	2DUP#=#	62CA5	ROTOVER	62F75	N+1DROP
628D1	2DUP#>	62CB9	DUPDUP	62F89	ROLLDROP
628EB	DUP#1+	62CCD	OVERDUP	62F9D	MDIMSDROP
62904	SWP1+	62CE1	COERCEDUP	62FB1	DUPROT
62904	SWAP#1+	62CF5	UNROTDUP	62FC5	DROPROT
6292F	DUP#1-	62D09	4UNROLLDUP	62FD9	#1-ROT
62946	DROPONE	62D1D	NTHCOMDDUP	62FED	%%*ROT
62958	RDROPCOLA	62D31	OVERUNROT	63001	FOURROLLROT
6296D	COLACOLA	62D31	OVERSWAP	63001	4ROLLROT
62986	COLAcase	62D45	ROLLSWAP	63015	4UNROLLROT
629A1	COLANOTcase	62D59	NULL\$SWAP	63029	DROPOVER
629BC	ORcase	62D6D	SUB\$SWAP	6303D	EQOVER
629D0	REQcase	62D81	%MAXorder	63051	#+OVER
629E9	REQcasedrop	62D9F	?SKIPSWAP	63065	#-OVER
62A02	SAFESTO	62DB3	1ABNSWAP	63079	ZEROOVER
62A2F	DUPSAFE@	62DCC	ROT+SWAP	6308D	UNROTOVER
62A34	SAFE@	62DCC	ROT#+SWAP	630A1	4ROLLOVER
62A61	?>ROMPTR	62DE5	4PICK+SWAP	630B5	3PICKOVER
62A84	?ROMPTR>	62DE5	4PICK#+SWAP	630C9	4PICKOVER
62ABB	MACRODCMP	62DFE	#+SWAP	630DD	DUPPICK
62B0B	2DROPFALSE	62E12	#-SWAP	630F1	DUPROLL
62B1F	PALPTRDCMP	62E26	#1+SWAP	63105	OVER#2+UNROLL
62B5B	palrompdcmp	62E3A	ZEROSWAP	63105	OVER#2+UNROL
62B6F	#0=UNTIL	62E4E	#1-1SWAP	63119	8UNROLL
62B88	INCOMPDROP	62E67	ONESWAP	6312D	10UNROLL
62B9C	NTHCOMPDROP	62E7B	COERCESWAP	63141	OVERARSIZE
62BB0	APPEND_SPACE	62E8F	%>%SWAP	63155	'ERRJMP
62BC4	7UNROLL	62EA3	%%*SWAP	63169	caseERRJMP
62BD8	RESOROMP	62EB7	XYZ>ZTRUE	6317D	?CARCOMP
62BF1	%10*	62ECB	4ROLLSWAP	63191	NEWLINE\$&&\$

63191	NEWLINE&\$	634B6	1GETABND	63768	ROT#-
631A5	#1-{}N	634CA	DUP1LAMBIND	6377C	OVER#-
631B9	TWO{}N	634CF	1LAMBIND	63790	INDEX@-
631CD	THREE{}N	634E3	caseTRUE	63790	INDEX@#-
631E1	DUPINCOMP	634F7	TRUEFALSE	637A4	SWAPOVER#-
631F5	SWAPINCOMP	634F7	TrueFalse	637B8	ROT#1+
63209	DUPNULL\$?	6350B	FALSETRUE	637CC	#--+1
6321D	DUPNULLCOMP?	6350B	FalseTrue	637CC	#1--
63231	DUPLENCOMP	6351F	ZEROFALSE	637E0	SWAP#1-
63245	#1-SUB\$	63533	ONEFALSE	637F4	DROP#1-
63259	1_#1-SUB	63547	#=casedrpfls	63808	#+-1
63259	1_#1-SUB\$	6356A	casedrpfls	63808	\$1-+
6326D	LAST\$	63583	case2drpfls	63808	#1-+
63281	#1+LAST\$	6359C	caseFALSE	6381C	COLAITE
63295	DUP\$>ID	635B0	ORNOT	6383A	ERROROUT
632A9	SWAP%>C%	635C4	EQUALNOT	6384E	DO=DSKTOP
632BD	'NOP	635D8	2DUPEQ	6385D	D1=DSKTOP
632D1	::NEVAL	635EC	DUPEQ:	6386C	SWAP2DUP
632E5	2GETEVAL	635F1	EQ:	63880	RSKIP
632F9	DROPRDROP	63605	EQOR	6389E	GROB!ZERODRP
63312	SWAPCOLA	63619	EQUALOR	638B2	casedrptru
63326	XYZ>ZCOLA	6362D	2#0=OR	638CB	NOTcaseTRUE
6333A	#0=?SKIP	6364B	OVER#0=	638E4	?SEMIDROP
63353	#1=?SKIP	6365F	OVER#<	638FD	SWAPUnDROP
6336C	#=?SKIP	63673	#<3	63911	SWAPUnNDROP
63385	ONE_EQ	63687	DUP#<7	63925	DUP'
63399	#>?SKIP	636A0	INNER#1=	63939	SWAP'
633B2	COLASKIP	636B4	#5=	6394D	DROP'
633C6	NOT_UNTIL	636C8	#2<>	63961	OVER'
633DF	NOT_WHILE	636DC	OVER#>	63975	STO'
633F8	DUP#0<>WHILE	636F0	ONE#>	63989	TRUE'
63411	DUPINDEX@	636F0	#>1	6399D	ONEFALSE'
63425	SWAPINDEX@	63704	DUP3PICK#+	639B6	FALSE'
63439	OVERINDEX@	63704	2DUP#+	639CA	#1+'
6344D	SWAPLOOP	63718	ROT#+	639DE	'R'R
63466	DROPLoop	6372C	OVER#+	639FC	'RRDROP
6347F	DUP#0_DO	63740	3PICK#+	63A15	ONECOLA
63498	toLEN_DO	63754	4PICK#+	63A29	dvarlsBIND

63A3D	'LAMLNAMESTO	63D26	#1=case	6400F	ZEROISTOPSTO
63A56	'xDEREQ	63D3A	#<>case	64023	RclHiddenVar
63A6F	DUPNULL{ }?	63D4E	#>2case	64037	WithHidden
63A88	DUPZERO	63D67	#>case	64078	StoHiddenVar
63A9C	DUPONE	63D7B	j%0=case	6408C	PuHiddenVar
63AB0	SWAPONONE	63D8F	REALcase	640A0	SaveVarRes
63AC4	ONEDUP	63DA3	dARRAYcase	640BE	SetHiddenRes
63AC4	ONEONE	63DB7	dLISTcase	640FA	RestVarRes
63AD8	DUPTWO	63DCB	EditExstCase	64127	Embedded?
63AEC	NOTcsdrpfls	63DDF	ANDNOTcase	6416D	UStackDepth
63B05	caseSIZEERR	63DF3	EQUALNOTcase	64190	Sig?ErrJump
63B19	NcaseSIZEERR	63E07	dIDNTNcase	641CC	DupAndThen
63B2D	CKREAL	63E1B	dREALNcase	641FC	ZEROZERO
63B46	NcaseTYPEERR	63E2F	EQIT	64209	#ZERO#ONE
63B5A	'x*	63E48	DUP#0=IT	6427A	#ZERO#SEVEN
63B6E	'xDER	63E61	ANDITE	6428A	#ONE#27
63B82	%/>%	63E75	EQITE	6429D	#TWO#ONE
63B96	UNCOERCE%%	63E89	#0=ITE	642AF	#TWO#TWO
63BAA	DUP%0=	63E9D	#<ITE	642BF	#TWO#FOUR
63BBE	SWAP%%/	63EB1	#>ITE	642D1	#THREE#FOUR
63BD2	caseDrpBadKy	63EC5	DUP#0=ITE	642E3	#FIVE#FOUR
63BEB	caseDEADKEY	63ED9	UserITE	64309	ZEROZEROZERO
63BEB	caseDoBadKey	63EED	SysITE	6431D	ZEROZEROONE
63C04	GROBDIMw	63F01	top&Cr	64331	ZEROZEROTWO
63C18	%*UNROT	63F1A	metaROTDUP	64345	SubMeta0b
63C2C	XYZW>YWZX	63F2E	ROUntop&	643BD	SubMeta0b1
63C2C	SWAP4ROLL	63F42	roll2top&	643EF	matchob?
63C40	2DUP5ROLL	63F42	rolltwotop&	643F9	matchob?Lp
63C54	SWAP3PICK	63F56	p1DRPpZparg	64426	POSCOMP
63C68	3PICK3PICK	63F6A	&\$SWAP	6443A	nextpos
63C7C	SWAP4PICK	63F7E	SWAPCKREF	64449	3DROPZERO
63C90	OVER5PICK	63F92	pZpargSWAPUn	6448A	#=POSCOMP
63CA4	EQUALcasedrp	63FA6	DROPNDROP	644A3	EQUALPOSCOMP
63CBD	DUP#0=csDROP	63FBA	2OVER	644BC	NTHOF
63CD6	jEQcase	63FCE	?0b>Seco	644D0	Find1stTrue
63CEA	ANDcase	63FE7	0b>Seco	644EE	Find1stT.1
63CFE	EQUALcase	63FFB	20b>Seco	6452F	Lookup
63D12	#<case	6400F	ExitAtLOOP	64548	Lookup.1

64593	EQLookup	64B8A	FIFTYSIX	64C3E	EIGHTYONE
645B1	POS\$	64B94	FIFTYSEVEN	64C3E	BINT81
645B1	POSCHR	64B94	BINT57	64C48	BINT82
645BD	POSCHRREV	64B9E	FIFTYEIGHT	64C48	LISTCMP
645BD	POS\$REV	64B9E	BINT58	64C52	BINT83
6475C	CHR>\$	64BA8	BINT59	64C52	FIVETHREE
64775	STRIPTAGS	64BA8	FIFTYNINE	64C5C	BINT84
647A2	STRIPTAGS12	64BB2	SIXTY	64C5C	FIVEFOUR
647BB	TAGOBS	64BB2	BINT60	64C66	BINT85
6480B	NEXTCOMPOB	64BBC	SIXTYONE	64C66	2LIST
648BD	>LASTRAM-WORD	64BBC	BINT61	64C70	BINT86
64B12	BINT44	64BC6	BINT62	64C70	FIVESIX
64B12	FORTYFOUR	64BC6	SIXTYTWO	64C7A	BINT87
64B1C	FORTYFIVE	64BD0	BINT63	64C7A	LISTLAM
64B1C	BINT45	64BD0	SIXTYTHREE	64C84	BINT_91d
64B26	BINT46	64BDA	SIXTYFOUR	64C84	BINT91
64B26	FORTYSIX	64BDA	BINT64	64C8E	BINT_96d
64B30	BINT47	64BDA	BINT40h	64C8E	BINT96
64B30	FORTYSEVEN	64BDA	YHI	64C98	BINT97
64B3A	BINT48	64BE4	BINT65	64C98	IDREAL
64B3A	FORTYEIGHT	64BE4	ARRYREAL	64CA2	BINT98
64B44	BINT49	64BEE	FOUR TWO	64CAC	ONEHUNDRED
64B44	FORTYNINE	64BEE	BINT66	64CAC	BINT100
64B4E	BINT50	64BF8	FOURTHREE	64CB6	BINT101
64B4E	FIFTY	64BF8	BINT67	64CC0	char
64B58	FIFTYONE	64C02	SIXTYEIGHT	64CC0	BINT111
64B58	BINT51	64C02	BINT68	64CCA	BINT112
64B62	BINT52	64C0C	BINT69	64CD4	BINT113
64B62	FIFTYTWO	64C0C	FOURFIVE	64CDE	BINT114
64B6C	BINT53	64C16	SEVENTY	64CE8	BINT_115d
64B6C	THREEFIVE	64C16	BINT70	64CE8	BINT115
64B6C	STRLIST	64C20	BINT74	64CF2	BINT_116d
64B6C	FIFTYTHREE	64C20	SEVENTYFOUR	64CF2	BINT116
64B76	FIFTYFOUR	64C2A	SEVENTYNINE	64CFC	BINT117
64B76	BINT54	64C2A	BINT79	64D06	BINT122
64B80	BINT55	64C3A	EIGHTY	64D06	BINT_122d
64B80	FIFTYFIVE	64C3A	BINT80	64D10	BINT80h
64B8A	BINT56	64C3E	LISTREAL	64D10	BINT128

64D1A	BINT130d	64E64	3REAL	64FE0	SYMREALSYM
64D1A	XHI-1	64E6E	Err#Kill	64FEA	SYMCMPPREAL
64D1A	BINT_130d	64E78	Err#NoLstStk	64FF4	SYMCMPCMP
64D1A	BINT130	64E82	#NoRoomForSt	64FFE	SYMCMPSYM
64D24	XHI	64E8C	#132	65008	SYMIDREAL
64D24	BINT131	64E96	REALSTRSTR	65012	SYMIDCMP
64D24	BINT131d	64EA0	#134	6501C	SYMIDLIST
64D24	BINT_131d	64EAA	#135	65026	SYMIDEXT
64D2E	#8F	64EB4	#136	65030	SYMSYMBOL
64D38	SYMBREAL	64EBE	#137	6503A	SYMSYMBOL
64D42	SYMBCMP	64EC8	#138	65044	3SYM
64D4C	SYMSYMBOL	64ED2	#139	6504E	XFERFAIL
64D56	SYMBUNIT	64EDC	#13A	65058	PROTERR
64D60	backup	64EE6	#13B	65062	InvalServCmd
64D6A	SYMBOL	64EF0	#13D	6506C	Connecting
64D74	SYMREAL	64EFA	Err#Cont	65076	Retry
64D88	SYMLIST	64F04	INTEGER337	65080	#CAlarmErr
64D92	SYMID	64F0E	CMPOBOB	6508A	EXTOBOB
64D9C	SYMLAM	64F18	Err#NoLstArg	65094	#EXITERR
64DA6	SYMSYMBOL	64F22	STRREALREAL	6509E	MINUSONE
64DB0	SYMSYMBOL	64F2C	ARRAYREALREAL	650A8	%e
64DBA	SYMEXT	64F36	ARRAYREALCMP	650BD	%.5
64DC4	HXSREAL	64F40	3ARRAY	650D2	%-.5
64DCE	2HXS	64F4A	ARRAYLISTREAL	650E7	%10
64DD8	BINTCOh	64F54	ARRAYLISTCMP	650FC	%180
64DE2	2GROB	64F5E	LISTREALOB	65111	%200
64DEC	TAGGEDANY	64F68	LISTREALREAL	65126	%360
64DF6	EXTREAL	64F72	LISTLISTOB	6513B	%400
64E00	EXTSYMBOL	64F7C	IDREALOB	65150	tok]
64E0A	2EXT	64F86	IDLISTOB	6515C	lbrac
64E14	ROMPANY	64F90	FSTMACROROM#	6516A	tok[
64E1E	BINT253	64F9A	PROGIDREAL	65176	tok{
64E28	BINT255d	64FA4	PROGIDCMP	65182	tok}
64E32	REALOBOB	64FAE	PROGIDLIST	6518E	toksharp
64E3C	#_102	64FB8	PROGIDEXT	6519A	tokuscore
64E46	#SyntaxErr	64FC2	ATTNERR	651A6	tok\$
64E50	BINT_263d	64FCC	SYMREALREAL	651B2	tok&
64E5A	REALREALOB	64FD6	SYMREALCMP	651BE	tokESC

651CA	tok>>	653B8	tok9	65521	CHR_M
651D6	tok<<	6541E	CHR_00	65528	CHR_N
651E2	tokexponent	65425	CHR_...	6552F	CHR_O
651EE	tokanglesign	6542C	CHR_DblQuote	65536	CHR_P
651FA	tokSIGMA	65433	CHR_#	6553D	CHR_Q
65206	tokWHERE	6543A	CHR_*	65544	CHR_R
65212	14SPACES\$	65441	CHR_+	6554B	CHR_S
65238	NEWLINE\$	65448	CHR_,	65552	CHR_T
65244	\$DER	6544F	CHR_-	65559	CHR_U
65254	tok_	65456	CHR_.	65560	CHR_V
65254	SPACE\$	6545D	CHR_/	65567	CHR_W
65260	tokUNKNOWN	65464	CHR_0	6556E	CHR_X
65278	tokquote	6546B	CHR_1	65575	CHR_Y
65284	tok'	65472	CHR_2	6557C	CHR_Z
65290	tok,	65479	CHR_3	65583	CHR_a
6529C	tok.	65480	CHR_4	6558A	CHR_b
652A8	tok;	65487	CHR_5	65591	CHR_c
652B4	toklparen	6548E	CHR_6	65598	CHR_d
652C0	tokrparen	65495	CHR_7	6559F	CHR_e
652CC	tok^	6549C	CHR_8	655A6	CHR_f
652D8	tok*	654A3	CHR_9	655AD	CHR_g
652E4	tok/	654AA	CHR_:	655B4	CHR_h
652F0	tok+	654B1	CHR_;	655BB	CHR_i
652FC	tok-	654B8	CHR_<	655C2	CHR_j
65308	tok=	654BF	CHR_=	655C9	CHR_k
65314	tokSQRT	654C6	CHR_>	655D0	CHR_l
65320	tokDER	654CD	CHR_A	655D7	CHR_m
6532C	tokCTGROB	654D4	CHR_B	655DE	CHR_n
6533E	tokCTSTR	654DB	CHR_C	655E5	CHR_o
6534C	tok0	654E2	CHR_D	655EC	CHR_p
65358	tok1	654E9	CHR_E	655F3	CHR_q
65364	tok2	654F0	CHR_F	655FA	CHR_r
65370	tok3	654F7	CHR_G	65601	CHR_s
6537C	tok4	654FE	CHR_H	65608	CHR_t
65388	tok5	65505	CHR_I	6560F	CHR_u
65394	tok6	6550C	CHR_J	65616	CHR_v
653A0	tok7	65513	CHR_K	6561D	CHR_w
653AC	tok8	6551A	CHR_L	65624	CHR_x

6562B	CHR_y	6595A	getnibs	7DD61	D/DLN
65632	CHR_z	659DE	Symb>HBuff	7DD6C	D/DLNP1
65639	CHR_->	66EA5	BigCursor	7DD77	D/DLOG
65640	CHR_<<	66ECD	MediumCursor	7DD82	D/DIFTE
65647	CHR_>>	66EF1	SmallCursor	7DD8D	D/DSIN
6564E	CHR_Angle	70601	BANKMTHDS	7DD98	D/DSINH
65655	CHR_Deriv	715B1	?ACCPTR>	7DDA3	D/DSQ
6565C	CHR_Integral	71C3B	rNTHELCOMP	7DDAE	D/DSQRT
65663	CHR_LeftPar	71DB2	RPTRACC	7DDB9	D/DTAN
6566A	CHR_Newline	7DBA0	nWHEREIFTE	7DDC4	D/DTANH
65671	CHR_Pi	7DBAB	nWHEREIDER	7DDCF	D/D^
65678	CHR_RightPar	7DBB6	nWHEREINTG	7DDDA	D/D^X
6567F	CHR_Sigma	7DBC1	nWHEREESUM	7DDE5	D/D^Y
65686	CHR_Space	7DBCC	nWHEREWHERE	7DDF0	D/DDER
6568D	CHR_UndScore	7DBD7	nWHEREFCNAPP	7DDFB	D/DWHERE
65694	CHR_ [7DBE2	D/D*	7DE06	D/DINTEGRAL
6569B	CHR_]	7DBED	D/D+	7DE11	D/DSUM
656A2	CHR_ {	7DBF8	D/D-	7DE1C	D/DAPPLY
656A9	CHR_ }	7DC03	D/D/	7DE27	nINTGINV
656B0	CHR_ <=	7DC0E	derquot	7DE32	nINTGSIGN
656B7	CHR_ >=	7DC54	derprod1	7DE3D	nINTGSQRT
656BE	CHR_ <>	7DC72	D/D=	7DE48	nINTGSQ
656C5	\$_R<<	7DC7D	D/DABS	7DE53	nINTGSIN
656D5	\$_R<Z	7DC88	easyabs	7DE5E	nINTGCOS
656E5	\$_XYZ	7DCA1	D/DACOS	7DE69	nINTGTAN
656F5	\$_<<>>	7DCAC	D/DACOSH	7DE74	nINTGSINH
65703	\$_{ }	7DCB7	D/DALOG	7DE7F	nINTGCOSH
65711	\$_ []	7DCC2	D/DARG	7DE8A	nINTGTANH
6571F	\$_ ' '	7DCCD	D/DASIN	7DE95	nINTGASIN
6572D	\$_ : :	7DCD8	D/DASINH	7DEA0	nINTGACOS
6573B	\$_LRParens	7DCE3	D/DATAN	7DEAB	nINTGATAN
65749	\$_2DQ	7DCEE	D/DATANH	7DEB6	nINTGLN
65757	\$_ECHO	7DCF9	D/DCHS	7DEC1	nINTGLOG
65769	\$_EXIT	7DD17	D/DCONJ	7DECC	nINTGALOG
6577B	\$_Undefined	7DD35	D/DCOS	7DED7	nINTGEXPM
65797	\$_RAD	7DD40	D/DCOSH	7DEE2	nCustomMenu
657A7	\$_GRAD	7DD4B	D/DEXP	7DEED	nCOLCTQUOTE
6594E	putnibs	7DD56	D/DINV	7DEF8	SPLITWHERE

7DF03	MANMENU+-	800EB	COVERsave	8065B	DISABLE_KBD
7DF0E	MANMENU*/	800F5	IRAMBUFF	8065B	HANDSHK
7DF19	MANMENU^	80127	IRAMBUFF2	8065C	KEYSTATE
7DF24	MANMENEUEXP	8030E	GraphPrtHook	80669	KEYBUFFER
7DF2F	MANMENULN	8030E	IRAMBEND	80669	INPUTSTREAM
7DF3A	MANMENUCSIV	80319	uart_buffer	8068B	POPPEDKEY
7DF45	MANMENEUEQ	80519	uart_buf_end	8068D	DISP1CTLg
7DF50	MANMENEUCX	8051B	uart_error	80692	LINENIBSg
7DF5B	MANMENUTRG	8051C	uart_buf_st	80695	DISP2CTLg
7DF66	MANMENUATG	8051E	uart_handshk	8069A	LINECOUNTg
7DF71	PTYPE>PINFO	8051F	uart_modes	8069C	Stk0save
7DF7C	MOVEVAR	80520	uart_parity	806A1	Stk1save
7DF87	COPYVAR	80521	uart_timeout	806A6	Stk2save
80000	RAMSTART	80523	IOCNIB	806AB	Stk3save
80000	CMOS	80524	CONFRAM	806B0	Stk4save
80000	HARDROMEND	8052B	CONFTAB	806B5	Stk5save
80005	IRAMMASK	80536	PORT0EOS	806BA	R2[A]save
8000A	HOMEMASK	8053B	PORT1EOS	806BF	R2[S]save
8000F	HARAMEND	80540	PORT2EOS	806C0	R1[A]save
80010	FAILSTK1	805DB	INTRAM	806C5	SAVE_BO
80022	FAILSTK2	805EB	SAVE_MODES	806C6	SAVE_LC
80034	FAILSTK3	805F0	SAVE_C[A]	806C8	SAVE_LN
80046	FAILSTK4	805F5	SAVE_A	806CB	SAVE_OFFSET
80058	NEXTIRQ	80605	SAVE_ST	806D0	VDISP2
80065	TIMECRC	80608	SAVE_B	806D5	ADISP
80069	TIMExmit	80618	SAVE_D	806DA	SYSUPSTART
80069	TIMEOUT	80628	SAVE_R0	806DA	VDISP
80076	TIMEOUTCLK	80638	SAVE_PC	806DA	VDISP1
80077	LoBatTime	8063D	SAVE_D0	806DF	VDISP3
80078	StartTime	80642	SAVE_OR	806E4	GDISP
80085	FailTime	80642	ORghost	806E9	TEMPOB
80092	TESTMSG	80645	DRSTART	806EE	TEMPTOP
800BE	SW_Image	8064A	DREND	806F3	RSKTOP
800D4	SW_ETime	8064F	IREG	806F8	DSKTOP
800E1	PortStat	80652	SEMAPH	806FD	EDITLINE
800E2	Port1CRC	80654	IOSAVE	80702	TEMPENV
800E6	AccessInit	80655	CSPEED	80707	DOLPENV
800E8	COVERstate	8065A	INITEN	8070C	TOUCHTAB

80711	USEROB	807C0	MenuKeyRS	80817	ITEM1LINES
80716	ROMPARTS	807C5	ReviewKey	80818	VIEWLEVEL
8071B	CONTEXT	807CA	LastContext	8081D	DEPTHSAVE
80720	STOPSIGN	807CF	TrackAct	80822	RNSEED
80725	UserKeys	807D4	MenuExitAct	80831	SAVECLK
8072A	ALARMS	807D9	LASTROMWDOB	80832	ALARMSDUE
8072F	INTRPPTR	807DE	KeyOb	80833	PASTDUE
8072F	OBUPSTART	807E8	SYSNOUNPSTART	80834	DOUSEALARM
80734	OSAVE	807E8	RAMEND	80835	NOALARMSRV
80739	LASTARG	807ED	AVMEM	80836	ALARM#
80739	LASTARG1	807F2	ERROR	8083B	PASTDUE#
8073E	LASTARG2	807F7	ATTNFLAG	80840	LPD_HIST
80743	LASTARG3	807FC	FIRSTPROC	80841	ANNUNCIATORS
80748	LASTARG4	80801	SysNib1	80843	SystemFlags
8074D	LASTARG5	80802	SysNib2	80853	UserFlags
80752	leeway	80803	SysNib3	80863	ELEMENT
80757	ITEM1STATE	80804	SysNib4	80865	FIRSTCHAR
8075C	HISTORY1	80805	SysNib5	8086A	CR_COUNT
80761	HISTORY2	80806	SysNib6	8086F	STACKNUM
80766	HISTORY3	80807	SysNib7	80874	TOPLINE
8076B	HISTORY4	80808	SysNib8	80879	LASTLAST?
80770	PDCHXS	80809	SysNib9	8087A	HISTORYLEVEL
80775	PDCSYMB	8080A	EDITFLAG	8087B	LASTARGCOUNT
80775	KERMERRM	8080A	SysNib10	8087C	LASTARGf
8077A	PAINTTREE	8080A	EDITLFLAG	8087D	LASTERROR
8077F	EXITMSG	8080B	ParenModFLAG	80882	CURSOREPOSN
80784	AppDisplay	8080B	SysNib11	80882	CURSOR
80789	AppKeys	8080C	SysNib12	80887	CURSORPART
8078E	AppExitCond	8080D	SysNib13	80887	CURSORROW
80793	AppError	8080E	SizeMLDisp	8088C	CURSORPOSN
80798	CtlAlarm	8080E	SysNib14	8088C	CURSOROFFSET
8079D	MenuDef	8080F	SysNib15	8088E	CURSORSTATE
807A2	LastMenuDef	80810	SysNib16	8088F	CURSORCHR
807A7	MenuData	80811	SysNib17	80891	CURSORGROB
807AC	MenuRowAct	80812	SysNib18	808B9	CURSORX
807B1	LabelDef	80813	SysNib19	808BE	CURSORY
807B6	MenuKeyNS	80814	SysNib20	808C3	CURRENTMENU
807BB	MenuKeyLS	80815	AppCount	808C5	MENULEVEL

808CA	OLDMENU	80928	LineByteCt	0230B3	~BBReReadPageSize
808CC	T1COUNT	8092A	FifoByteCt	0240B3	~BBReReadHeight
808CD	PADCOUNT	8092C	LastPrntTime	0250B3	~BBReReadCoords
808CE	GARBSCRATCH1	80937	PFIFO	0260B3	~BBReReadWidth
808D3	GARBSCRATCH2	80947	MenuRow	0280B3	~BBRunENTERAction
808D8	SAVECROSS	8094C	EqPtr	0290B3	~BBRunCanclAction
808E2	PADJSAVE1	80951	KeyRomPtr0	02F0B3	~BBReDrawBackgr
808E3	PADJSAVE2	8095C	KeyRomPtr1	0370B3	~BBGetNGrob
808ED	KERMMODE	80967	KeyRomPtr2	0380B3	~BBGetNStr
808EE	DcompWidth	80972	KeyRomPtr3	03B0B3	~BBRereadChkEnbl
808F0	FONTHEIGHT	8097D	KeyRomPtr4	03C0B3	~BBRereadFullScr
808F1	FONTWIDTH	80988	KeyRomPtr5	03D0B3	~BRReReadMenus
808F2	FONTCOUNT	80993	KeyRomPtr6	03E0B3	~BBReReadNElems
808F5	NODECOUNT	8099E	LastMenuRow	03F0B3	~BBGetN
808F8	OBTREELEN	809A3	RESRAMEND	04B0B3	~BBIsChecked?
808FB	LASTOP	809A3	ROMPTAB	0520B3	~BBUpArrow
808FC	LEFTTREE	809A3	FlashROMPTAB	0530B3	~BBDownArrow
808FF	RIGHTTREE	90000	HARDDRAMEND	0540B3	~BBSpace
80902	PARENTTREE	9358F	TYPEARRY@	0590B3	~BBPgDown
80905	PRECSTACK	0050B0	~IFMenuRow1	05A0B3	~BBPgUp
8090C	KEYLIST	0060B0	~IFMenuRow2	05B0B3	~BBEmpty?
80911	KEYLOCK	0850B0	~grobAlertIcon	05C0B3	~BBGetDeflhtHeight
80912	ACCUM	0860B0	~grobCheckKey	05F0B3	~\$>grobOrGROB
80913	OB/EXP?	0C50B0	~gFldVal	0660B3	~ChooseSimple
80914	COLWIDTH	0D50B0	~sFldVal	0030E0	~BRStoC1
80916	ENTRWISE	0DB0B0	~nNullBind	0100E0	~BRbrowse
80917	PARENCOUNT	0000B1	~DoMsgBox	0120E0	~BRroutput
80919	STRETCHCOUNT	0020B1	~MsgBoxMenu	0170E0	~BRRc1CurRow
8091B	ClkOnNib	0000B3	~Choose	0180E0	~BRRc1C1
8091C	XmitSrcvTOut	0050B3	~ChooseMenu0	0630E3	~PCunpack
8091E	DelayCt	0060B3	~ChooseMenu1	0120E4	~MESRc1Eqn
80920	GCOLCOUNT	0070B3	~ChooseMenu2	0110E7	~UTVUNS1Arg
80922	COLCOUNT	0150B3	~BBMoveTo	0580E7	~UTTYPEEXT0?
80924	PrtStatus	0190B3	~BBRecalOff&Disp	01E0E8	~INTEMPOB?
80927	IOCSave	0220B3	~BBRunEntryProc		

Entry Index

!			
!!append\$	34	#>	16
!!append\$?	34	#>\$	32
!!insert\$	34	#>%	38
!#1-IF>0	282	#>?SKIP	91
!#1+IF<dim-1	282	#>1	16
!*triand	36	#>2case	92
!*trior	36	#>case	91
!AND\$	35	#>CHR	32
!append\$	34	#>HXS	38
!append\$SWAP	34	#>ITE	91
!CHANGETYPE	113	#>ROMPTR	72
!DcompWidth	36	#<	16
!insert\$	34	#<>	16
!MAKEHXS	38	#<>case	91
!NOT\$	35	#<3	16
!OR\$	35	#<case	91
!TurnOff	112	#<ITE	91
!XOR\$	35	#0=	16
		#0=?SEMI	91
		#0=?SKIP	91
		#0=case	92
		#0=ITE	91
		#0=UNTIL	100
#		#0<>	16
##	14	#1	14
##*OVF	14	#1--	14
##-	14	#1-{}N	51
##-#2/	14	#1-+	14
##+1	14	#1-1SWAP	15
##-DUP	15	#1-DUP	15
##-OVER	15	#1-ROT	15
##-PICK	79	#1-SUB\$	34
##-ROLL	78	#1-SWAP	15, 53
##-SWAP	15	#1-UNROT	15
##-UNROLL	79	#1=	16
##/	14	#1=?SKIP	92
##:>\$	32	#1=case	92
##=	16	#1+	14
##=?SKIP	91	#1+'	98
##=case	91	#1+_ONE_DO	101
##=casedrop	91	#1+DUP	15
##=casedrpfls	91	#1+LAST\$	34
##=ITE	91	#1+NDROP	52, 76
##=POSCOMP	49	#1+PICK	79
##_102	10	#1+ROLL	78
##+	14	#1+ROT	15
##+-1	14	#1+SWAP	15
##+DUP	14	#1+UNROLL	79
##+OVER	15	#1<>	16
##+OVF	14	#10*	14
##+PICK	79	#10+	14
##+ROLL	78	#12+	14
##+SWAP	14		
##+UNROLL	79		

#132 11
 #134 11
 #135 11
 #136 11
 #137 11
 #138 11
 #139 11
 #13A 11
 #13B 11
 #13D 11
 #2* 14
 #2- 14
 #2/ 14
 #2= 16
 #2+ 14
 #2+PICK 79
 #2+ROLL 78
 #2+UNROLL 79
 #2<> 16
 #3- 14
 #3= 16
 #3+ 14
 #3+PICK 79
 #4- 14
 #4+ 14
 #4+PICK 79
 #5- 14
 #5= 16
 #5+ 14
 #6* 14
 #6- 14
 #6+ 14
 #7+ 14
 #8* 14
 #8+ 14
 #8F 9
 #9+ 14
 #AND 16
 #CAlarmErr 12
 #EXITERR 12
 #FIVE#FOUR 13
 #MAX 16
 #MIN 15
 #NoRoomForSt 10
 #ONE#27 12
 #SyntaxErr 10
 #THREE#FOUR 13
 #TWO#FOUR 13
 #TWO#ONE 13
 #TWO#TWO 13
 #ZERO#ONE 12
 #ZERO#SEVEN 12

\$

\$_ , ' 30
 \$_ : 30
 \$_ :: 30
 \$_ [] 30
 \$_ {} 30
 \$_ <<>> 30
 \$_2DQ 30
 \$_ECHO 31
 \$_EXIT 31
 \$_GRAD 31
 \$_LRParens 30
 \$_R<< 31
 \$_R<Z 31
 \$_RAD 31
 \$_Undefined 31
 \$_XYZ 31
 \$>\$? 37
 \$>=? 38
 \$>BIGGROB 156
 \$>grob 156
 \$>GROB 156
 \$>grobOrGROB (~\$>grobOrGROB) 156
 \$>ID 80
 \$<\$? 37
 \$<=? 38
 \$1-+ 14
 \$5x7 262
 \$DER 31

%

%%* 22
 %%*ROT 22
 %%*SWAP 22
 %%*UNROT 22
 %%- 22
 %%.1 18
 %%.4 18
 %%.5 18
 %%/ 22
 %%/>% 22
 %%+ 22
 %%> 23
 %%>% 19
 %%>= 23
 %%>C% 24
 %%>C%% 25
 %%^ 22
 %%< 23
 %%<= 23

%%0	18	%-1	17
%%0=	23	%-1=case	93
%%0>	24	%-2	17
%%0>=	24	%-3	17
%%0<	24	%-4	17
%%0<=	24	%-5	17
%%0<>	23	%-6	17
%%1	18	%-7	17
%%1/	22	%-8	17
%%10	19	%-9	17
%%12	19	%-MAXREAL	17
%%2	19	%-MINREAL	17
%%2PI	19	% .1	17
%%3	19	% .15	17
%%4	19	% .5	17
%%5	19	%/	20
%%60	19	%=	23
%%7	19	%+	20
%%ABS	22	%+SWAP	20
%%ACOSRAD	22	%>	23
%%ANGLE	22	%>#	38
%%ANGLEDEG	22	%>%%	19
%%ANGLERAD	22	%>%%-	20
%%ASINRAD	22	%>%%1/	20
%%CHS	22	%>%%ANGLE	21
%%COS	22	%>%%SQRT	20
%%COSDEG	23	%>%%SWAP	19
%%COSH	23	%>=	23
%%COSRAD	23	%>C%	24
%%EXP	23	%>HMS	107
%%FLOOR	23	%>TAG	40
%%H>HMS	107	%^	20
%%INT	23	%<	23
%%KZERO	19	%<=	23
%%LN	23	%<>	23
%%LNP1	23	%0	17
%%MAX	23	%0=	23
%%P>R	23	%0=case	92
%%PI	19	%0>	23
%%R>P	23	%0>=	23
%%RZERO	19	%0<	23
%%SIN	23	%0<>	23, 87
%%SINDEG	23	%1	17
%%SINH	23	%1-	20
%%SINRAD	22	%1/	20
%%SQR	22	%1=case	92
%%SQRT	23	%1+	20
%%SUM	68	%10	17
%%TANRAD	23	%10*	20
%*	20	%100	18
%-	20	%11	17
%-.5	17	%12	17

%1200	18	%D>R	22
%13	17	%e	17
%14	17	%EXP	20
%15	17	%EXPM1	20
%16	17	%EXPONENT	21
%17	18	%FACT	21
%18	18	%FLOOR	21
%180	18	%FP	21
%19	18	%HMS-	107
%1pshm1	54	%HMS+	107
%2	17	%HMS>	107
%2=case	93	%IFTE	68
%20	18	%INT	21
%200	18	%IP	21
%21	18	%IP>#	19
%22	18	%LN	20
%23	18	%LNP1	20
%24	18	%LOG	20
%2400	18	%MANTISSA	21
%25	18	%MAX	21
%26	18	%MAXIMIZE	21
%27	18	%MAXorder	21
%2PI	17	%MAXREAL	18
%3	17	%MIN	21
%360	18	%MINREAL	17
%4	17	%MOD	21
%400	18	%NFACT	21
%4800	18	%NROOT	21
%5	17	%OF	22
%6	17	%PERM	21
%7	17	%PI	17
%8	17	%POL>%REC	22
%80	18	%R>D	22
%9	17	%RAN	21
%9600	18	%RANDOMIZE	21
%ABS	20	%REC>%POL	22
%ABSCOERCE	13	%SGN	20
%ACOS	20	%SIN	20
%ACOSH	21	%SINH	21
%ALOG	20	%SPH>%REC	22
%ANGLE	21	%SQ	20
%ARG	21	%SQRT	20
%ASIN	20	%SYM%CH	60
%ASINH	21	%SYM%OF	60
%ATAN	21	%SYM%T	60
%ATANH	21	%SYM>	62
%CEIL	21	%SYM>=	62
%CH	22	%SYM<	62
%CHS	20	%SYM<=	62
%COMB	21	%SYMAND	61
%COS	20	%SYMCOMB	60
%COSH	21	%SYMMAX	59

%SYMMIN	59
%SYMMOD	59
%SYMOR	61
%SYMPERM	60
%SYMSUM	68
%SYMXOR	61
%SYMXROOT	60
%T	22
%TAN	20
%TANH	21
%TICKSday	18
&	
&\$	34
&\$SWAP	34
&COMP	48
&HXS	38
,	
'	98
'DoBadKey	98
'DoBadKeyT	98
'DROPFALSE	98
'ERRJMP	98
'EvalNoCK:_sup	121
'IDBAR	99
'IDCONIC	99
'IDFUNCTION	99
'IDHISTOGRAM	99
'IDPAR	80, 156
'IDPARAMETER	99
'IDPOLAR	99
'IDSCATTER	99
'IDTRUTH	99
'idUserKeys	80
'idUserKeys.	80
'IDX	80
'LamKPSto	114
'LAMLNAMESTO	103
'NOP	98
'R	96
'R'R	96
'Rapndit	99
'REVAL	96
'RRDROP	96
'RSaveRomWrd	121
'RSAVEWORD	121
'x*	99
'xDER	99
'xDEREQ	99
:	
::N	50
::NEVAL	52
?	
?>ROMPTR	73
?ACCPTR>	282
?addinver:	55
?addrever	55
?addsimir	55
?ATTN_QUIT	125
?ATTNQUIT	125
?BlinkCursor	114
?CARCOMP	48
?CaseKeyDef	92
?CaseRomptr@	92
?ClrAlg	148
?ClrAlgSetPr	148
?DispMenu	130, 143
?DispStack	143
?DispStatus	143
?expan^	64
?expanapp	64
?expaneginv	64
?FlashAlert	113
?GOTO	97
?Key>UKeyOb	127
?Ob>Seco	52
?OKINALG	123
?PURGE_HERE	103
?RollUpDA2	282
?ROMPTR>	73
?SEMI	89
?SEMIDROP	89
?SetEditRoll	146
?SKIP	89
?SKIPSWAP	89
?spliteq	70
?STO_HERE	103
?SWAP	89
?SWAPDROP	89
?TogU/LCase	138
@	
@	102
@LAM	81
@LAMNOT	102
@OBNOT	102

{	
{>TAG	40
{>N	50
+	
+LOOP	101
>	
>ALRMLS	109
>ARRY	44
>BAK	75
>DATE	107
>DelKey	140
>dvars	55
>H\$	35
>HCOMP	48
>LASTRAM-WORD	282
>MENU	131
>R	97
>Review\$	37
>SkipKey	141
>T\$	35
>TAG	40
>TCOMP	48
>TIME	107
>TOPTMP	106
<	
<DelKey	140
<SkipKey	140
0	
OLASTOWDOB!	121
OLastRomWrd!	121
1	
1/X15	282
1_#1-SUB	34
1_#1-SUB\$	34
10GETLAM	82
10PUTLAM	82
10UNROLL	78
11GETLAM	82
11PUTLAM	82
12GETLAM	82
12PUTLAM	82
13GETLAM	82
13PUTLAM	82
14GETLAM	82
14PUTLAM	82
14SPACES\$	30
15GETLAM	82
15PUTLAM	82
16GETLAM	82
16PUTLAM	82
17GETLAM	82
17PUTLAM	82
18GETLAM	82
18PUTLAM	82
19GETLAM	82
19PUTLAM	83
1A/LockA	112
1ABND\$WAP	83
1GETABND	83
1GETLAM	81
1GETLAMSWP1+	83
1GETSWAP	83
1LAMBIND	80
1NULLLAM{}	83
1PDCMASK	273
1POLKeyUI	136
1PUTLAM	82
1REV	18
1stkdecomp\$w	37
2	
2#0=OR	16
2%>%	19
2%>%	19
2%>%SQR	22
2'RCOLARPITE	89
2@REVAL	97
2OGETLAM	82
2OPUTLAM	83
21GETLAM	82
21PUTLAM	83
22PUTLAM	83
2DROP	76
2DROP00	13
2DROP2dropf	56
2DropBadKey	126
2DROPPFALSE	87
2DROPTTRUE	87
2DUP	76
2DUP#=#	16
2DUP#+	15
2DUP#>	16
2DUP#<	16

2DUP%R	22
2DUP5ROLL	76
2DUPEQ	88
2DUPSWAP	76
2EXT	10
2GETEVAL	83
2GETLAM	81
2GROB	10
2HXS	10
2HXS>#	13
2HXSLIST?	13
2LEN\$#+	33
2LIST	9
2NULLLAM{}	83
2Ob>Seco	52
2OVER	79
2psh1&rev:	54
2pull2DROP	54
2PUTLAM	82
2RDROP	97
2REAL	6
2Repl-1	56
2Repl0	56
2SINNER	56
2SINNERtop&	56
2SKIP	99
2skipcola	100
2SWAP	77
2top&	53

3

3@REVAL	97
3ARRY	11
3DROP	76
3DROPTRUE	87
3DROPZERO	13
3GETLAM	81
3NULLLAM{}	83
3PICK	79
3PICK#+	15
3PICK3PICK	79
3PICKOVER	79
3PICKSWAP	79
3PUTLAM	82
3RDROP	97
3REAL	10
3SKIP	99
3skipcola	100
3SYM	12
3UNROLL	78

4

4DROP	76
4DROP%O	19
4DROPFALSE	87
4DropLoop	263
4DROPTRUE	87
4GETLAM	82
4NULLLAM{}	83
4PICK	79
4PICK#+	15
4PICK#+SWAP	15
4PICK+SWAP	15
4PICKOVER	79
4PICKSWAP	79
4psh	54
4PUTLAM	82
4roll	52
4ROLL	77
4ROLLDROP	77
4ROLLOVER	77
4ROLLROT	77
4ROLLSWAP	77
4rollunrot	53
4unroll	52
4UNROLL	78
4UNROLL3DROP	78
4UNROLLDUP	78
4UNROLLROT	78

5

5DROP	76
5DROPFALSE	87
5GETLAM	82
5PICK	79
5PUTLAM	82
5roll	52
5ROLL	78
5ROLLDROP	78
5skipcola	100
5unroll	52
5UNROLL	78

6

6DROP	76
6GETLAM	82
6PICK	79
6PUTLAM	82
6ROLL	78
6UNROLL	78

7

7DROP	76
7GETLAM	82
7NULLLAM{}	83
7PICK	79
7PUTLAM	82
7ROLL	78
7UNROLL	78

8

8GETLAM	82
8NULLLAM{}	83
8PICK	79
8PUTLAM	82
8ROLL	78
8UNROLL	78

9

9GETLAM	82
9PUTLAM	82

A

a%>\$	32
a%>\$,	32
AbbrevStack?	282
AbbrStkMASK	272
ABND	81
ABORT	84
ABUFF	141
AccessInit	277
ACCUM	276
ACK_INIT	117
ACKALLALMS	109
ACKALM	109
ADDF	282
addrADISP	282
addrATTNFLAG	282
addrClkOnNib	282
addrKEYSTATE	282
addrLINECNTg	282
addrORghost	282
addrTEMPENV	282
addrTEMPTOP	282
addrVDISP	282
addrVDISP2	282
addt*	68
addt-	68
addt:	54
addt+	68
addt2:	54
addtics	282
addtINV	68
addtNEG	68
addtpsh	54
ADISP	270
AdjEdModes	282
ADJMEM	258
adr_uart_han	282
adrDISABLE_K	282
adrGraphPrth	282
adrKEYBUFFER	282
adrTIMEOUTCLK	282
AEQ1stcase	93
AEQopscase	93
AGAIN	100
aH>HMS	282
AINRTN	256
ALARM#	277
ALARM?	109
ALARMS	276
ALARMS@	108
ALARMSDUE	276
ALARMxcp	282
Alert\$	113
AlertStatus	113
AlgEntry?	148
AlgEntryStat	282
ALGeq?	282
ALGMASK	272
allkeys	283
AllowIntr	256
AllowPrldCl	116
ALRMLS>	109
aMODF	283
AND	88
AND\$	35
ANDcase	90
ANDITE	89
ANDNOTcase	90
AngleStatus	283
Ann_Alert.b	270
Ann_Alpha.b	270
Ann_ALT.b	271
Ann_Busy.b	270
Ann_IO.b	270
Ann_RAD.b	270
Ann_Shift.b	270
Ann_Susp.b	270
ANNCTRL	261
ANNCTRL2	261
ANNUNCIATORS	270

any 5
 APNDCRLF 33, 115
 apndvarlst 51
 AppCount 275
 AppDisplay 275
 AppDisplay! 136
 AppDisplay@ 136
 APPEND_SPACE 35
 AppError 275
 AppError! 137
 AppError? 137
 AppError@ 137
 AppExitCond 276
 AppExitCond! 136
 AppExitCond@ 137
 AppKeys 276
 AppKeys! 136
 AppKeys@ 136
 AppKeys0 136
 AppMode? 137
 AppModeMASK 273
 APPprompt1! 283
 APPprompt2 283
 AppSuspOK? 137
 ARGNUMERR 86
 argtypeerr 256
 argum 52
 argvalerr 256
 array 5
 ARRYEL? 41
 ARRYLISTCMP 11
 ARRYLISTREAL 11
 ARRYREAL 8
 ARRYREALCMP 11
 ARRYREALREAL 11
 ARSIZE 41
 ASLW5 266
 ASRW5 266
 ASuspOKMASK 272
 Attn# 11
 ATTN# 11
 Attn? 264
 ATTN? 125
 ATTNchk 265
 ATTNERR 11
 ATTNFLG 277
 ATTNFLG@ 125
 ATTNFLGCLR 125
 ATTNxcp 283
 AtUserStack 121
 AUTOSCALE 159
 AVMEM 270

B

B%0= 27
 B%ABS 27
 B%NEG 27
 backup 10
 BadMenuMASK 272
 BadPOLUIMASK 273
 BadTOLUIMASK 274
 BAK>OB 75
 BAKNAME 75
 BANGARRY 41
 BANKMTHDS 283
 BASE 111
 BASECHAR 111
 BBDownArrow (~BBDownArrow) 135
 BBEmpty? (~BBEmpty?) 135
 BBGetDefltHeight (~BBGetDefltHeight) 135
 BBGetN (~BBGetN) 135
 BBGetNGrob (~BBGetNGrob) 134
 BBGetNStr (~BBGetNStr) 134
 BBIsChecked? (~BBIsChecked?) 135
 BBMoveTo (~BBMoveTo) 133
 BBox/StdLbl: 155
 BBPgDown (~BBPgDown) 135
 BBPgUp (~BBPgUp) 135
 BBRecalOff&Disp (~BBRecalOff&Disp) 133
 BBReDrawBackgr (~BBReDrawBackgr) 134
 BBReReadChkEnbl (~BBReReadChkEnbl) 134
 BBReReadCoords (~BBReReadCoords) 133
 BBReReadFullScr (~BBReReadFullScr) 134
 BBReReadHeight (~BBReReadHeight) 133
 BBReReadNElems (~BBReReadNElems) 134
 BBReReadPageSize (~BBReReadPageSize) 133
 BBReReadWidth (~BBReReadWidth) 134
 BBRunCanclAction (~BBRunCanclAction) 134
 BBRunENTERAction (~BBRunENTERAction) 134
 BBRunEntryProc (~BBRunEntryProc) 133
 BBSpace (~BBSpace) 135
 BBUpArrow (~BBUpArrow) 135
 BEGIN 100
 BigCursor 152
 BIGDISPN 151
 BIGDISPROW1 150
 BIGDISPROW2 151
 BIGDISPROW3 151
 BIGDISPROW4 151
 BIND 80
 BindMatVars 283
 BINT_115d 9
 BINT_116d 9
 BINT_117h 10
 BINT_122d 9

BINT_130d.....	9	BINT35.....	6
BINT_131d.....	9	BINT36.....	6
BINT_263d.....	10	BINT37.....	7
BINT_91d.....	9	BINT38.....	7
BINT_96d.....	9	BINT39.....	7
BINT_AFh.....	10	BINT4.....	5
BINT0.....	5	BINT40.....	7
BINT1.....	5	BINT40h.....	8
BINT10.....	5	BINT41.....	7
BINT100.....	9	BINT42.....	7
BINT101.....	9	BINT43.....	7
BINT11.....	5	BINT44.....	7
BINT111.....	9	BINT45.....	7
BINT112.....	9	BINT46.....	7
BINT113.....	9	BINT47.....	7
BINT114.....	9	BINT48.....	7
BINT115.....	9	BINT49.....	7
BINT116.....	9	BINT5.....	5
BINT117.....	9	BINT50.....	7
BINT12.....	5	BINT51.....	7
BINT122.....	9	BINT52.....	7
BINT128.....	9	BINT53.....	7
BINT13.....	5	BINT54.....	7
BINT130.....	9	BINT55.....	7
BINT130d.....	9	BINT56.....	7
BINT131.....	9	BINT57.....	8
BINT131d.....	9	BINT58.....	8
BINT14.....	5	BINT59.....	8
BINT15.....	5	BINT6.....	5
BINT16.....	5	BINT60.....	8
BINT17.....	6	BINT61.....	8
BINT18.....	6	BINT62.....	8
BINT19.....	6	BINT63.....	8
BINT2.....	5	BINT64.....	8
BINT20.....	6	BINT65.....	8
BINT21.....	6	BINT66.....	8
BINT22.....	6	BINT67.....	8
BINT23.....	6	BINT68.....	8
BINT24.....	6	BINT69.....	8
BINT25.....	6	BINT7.....	5
BINT253.....	10	BINT70.....	8
BINT255d.....	10	BINT74.....	8
BINT26.....	6	BINT79.....	8
BINT27.....	6	BINT8.....	5
BINT28.....	6	BINT80.....	8
BINT29.....	6	BINT80h.....	9
BINT3.....	5	BINT81.....	8
BINT30.....	6	BINT82.....	8
BINT31.....	6	BINT83.....	8
BINT32.....	6	BINT84.....	9
BINT33.....	6	BINT85.....	9
BINT34.....	6	BINT86.....	9

BINT87	9	BRRc1C1 (~BRRc1C1)	135
BINT9	5	BRRc1CurRow (~BRRc1CurRow)	135
BINT91	9	BRStoC1 (~BRStoC1)	135
BINT96	9	BUILDKPACKET	283
BINT97	9		
BINT98	9		
BINTCOh	10		
bit#%*	39	C	
bit#%-	39	C%>%	24
bit#%/	39	C%>C%	24
bit#%+	39	C%0=	26
bit%#*	39	C%1	24
bit%#-	39	C%ABS	26
bit%#/#	39	C%*C*	26
bit%#+	39	C%*C*R	26
bit*	39	C%*C-C	26
bit-	39	C%*C-R	26
bit/	39	C%*C/C	26
bit+	38	C%*C/R	26
bitAND	39	C%*C+C	26
bitASR	39	C%*C+R	26
BITMAP	283	C%*CHS	26
bitNEG	40	C%*CONJ	26
bitNOT	39	C%*R*C	26
bitOR	39	C%*R-C	26
bitRL	39	C%*R/C	26
bitRLB	39	C%*R+C	26
bitRR	39	C%*i	24
bitRRB	39	C%-1	24
bitSL	39	C%-1=case	93
bitSLB	39	C%-i	24
bitSR	39	C%>#	13
bitSRB	40	C%>%	19
bitXOR	39	C%>%%	24
Blank\$	33	C%>%%SWAP	24
Blank&GROB!	151	C%0	24
BlankDA1	147	C%0=	26
BlankDA12	147	C%0=case	92
BlankDA2	147	C%1	24
BlankDA2a	147	C%1/	25
BLANKIT	147	C%1=case	92
BLINKMASK	272	C%2=case	93
BodyMore?	64	C%ABS	25
BOTROW	149	C%ACOS	25
Box/StdLabel	155	C%ACOSH	25
Box/StdLbl:	155	C%ALOG	25
BoxLabelGrob	152	C%ARG	25
BRbrowse (~BRbrowse)	135	C%ASIN	25
BReReadMenus (~BReReadMenus)	134	C%ASINH	25
BROADENGROB	142	C%ATAN	25
BROADENHBUFF	142	C%ATANH	25
BRoutput (~BRoutput)	135	C%*C*	24, 25
		C%*C*R	25

C%C-C	24	CDRCOMP	48
C%C-R	25	CENTER\$3x5	156
C%C/C	25	cfC	18
C%C/R	25	cfF	18
C%C+C	24	CHANGETYPE	113
C%C+R	25	char	9
C%C^C	25	Char>Edit	139
C%C^R	25	CharEdit	139
C%CHS	25	check_pdata	283
C%CONJ	25	CHECKHEIGHT	153
C%COS	25	CHECKKEY	124
C%COSH	25	CHECKMENU	131
C%EXP	25	CheckMenuRow	131
C%i	24	CHECKPICT	156
C%LN	25	CHECKPVAR	157
C%LOG	25	ChkGrHook	283
C%R*C	25	ChkLowBat	112
C%R-C	25	Choose (~Choose)	132
C%R/C	25	CHOOSE_INIT	117
C%R+C	25	ChooseMenu0 (~ChooseMenu0)	133
C%R^C	25	ChooseMenu1 (~ChooseMenu1)	133
C%SGN	25	ChooseMenu2 (~ChooseMenu2)	133
C%SIN	25	ChooseSimple (~ChooseSimple)	133
C%SINH	25	CHR_#	27
C%SQRT	25	CHR_*	27
C%TAN	25	CHR_.	27
C%TANH	25	CHR_.	27
C/i	24	CHR_-	27
C>Im%	24	CHR_->	29
C>Re%	24	CHR_.	27
CACHE	81	CHR_.	27
CacheStack	81	CHR_/	27
CALCINTEG	65	CHR_:	27
CAR\$	33	CHR_;	27
CARCOMP	48	CHR_=	28
case	89	CHR_[28
case2drop	90	CHR_]	28
case2DROP	90	CHR_{	29
case2drpf1s	90	CHR_}	29
caseDEADKEY	96	CHR_+	27
caseDoBadKey	96	CHR_>	28
casedrop	90	CHR_>=	29
caseDROP	90	CHR_>>	29
caseDrpBadKy	96	CHR_<	28
casedrpf1s	90	CHR_<=	29
casedrptru	90	CHR_<>	29
caseERRJMP	96	CHR_<<	29
caseFALSE	90	CHR_0	27
caseSIZEERR	96	CHR_00	27
caseTRUE	90	CHR_1	27
CCSB1	266	CHR_2	27
CDR\$	33	CHR_3	27
		CHR_4	27

CHR_5	27	CHR_Space	27
CHR_6	27	CHR_t	29
CHR_7	27	CHR_T	28
CHR_8	27	CHR_u	29
CHR_9	27	CHR_U	28
CHR_a	28	CHR_UndScore	28
CHR_A	28	CHR_v	29
CHR_Angle	29	CHR_V	28
CHR_b	28	CHR_w	29
CHR_B	28	CHR_W	28
CHR_c	28	CHR_x	29
CHR_C	28	CHR_X	28
CHR_d	28	CHR_y	29
CHR_D	28	CHR_Y	28
CHR_DblQuote	27	CHR_z	29
CHR_Deriv	29	CHR_Z	28
CHR_e	28	CHR>#	13
CHR_E	28	CHR>\$	32
CHR_f	28	CINRTN	256
CHR_F	28	CK%ACOS	20
CHR_g	29	CK%ACOSH	21
CHR_G	28	CK%ASIN	20
CHR_h	29	CK%ATANH	21
CHR_H	28	CK%LN	20
CHR_i	29	CK%LOG	20
CHR_I	28	CK%SQRT	20
CHR_Integral	29	Ck&AsnUKeys	126
CHR_j	29	Ck&ClrUKey	127
CHR_J	28	Ck&DecKeyLoc	123
CHR_k	29	CK&DISPATCHO	120
CHR_K	28	CK&DISPATCH1	120
CHR_l	29	CK&DISPATCH2	120
CHR_L	28	Ck&Freeze	146
CHR_LeftPar	27	CKO	119
CHR_m	29	CKOATTNABORT	125
CHR_M	28	CKONOLASTWD	119
CHR_n	29	CK1	119
CHR_N	28	CK1&Dispatch	120
CHR_Newline	27	CK1NoBlame	121
CHR_o	29	CK1NOLASTWD	119
CHR_O	28	CK2	119
CHR_p	29	CK2&Dispatch	120
CHR_P	28	CK2NOLASTWD	119
CHR_Pi	29	CK3	119
CHR_q	29	CK3&Dispatch	120
CHR_Q	28	CK3NOLASTWD	119
CHR_r	29	CK4	119
CHR_R	28	CK4&Dispatch	120
CHR_RightPar	27	CK4NOLASTWD	119
CHR_s	29	CK5	119
CHR_S	28	CK5&Dispatch	120
CHR_Sigma	29	CK5NOLASTWD	119

CkChr00	38	ClrDA1ValidF	145
CkEQUtil	283	ClrDA2aBad	145
ckevaleq1	58	ClrDA2aOK	143
CKGROBFITS	153	ClrDA2aTempF	144
CKINFARGS	121	ClrDA2aValidF	145
CKLBCRC	260	ClrDA2bBad	145
CKN	119	ClrDA2bEdit	146
CKNFLOATS	121	ClrDA2bIsEdL	146
CKNNOLASTWD	120	ClrDA2bNoCh	146
CkNonNull	104	ClrDA2bOK	143
cknseval:	58	ClrDA2bTemp	144
cknum2:	58	ClrDA2bTempF	144
cknumdsptch1	57	ClrDA2bValidF	145
CKPICT	156	ClrDA2NoCh	146
CKREAL	121	ClrDA2OK	143
CKREF	106	ClrDA3Bad	145
CkSecoType	283	ClrDA3NoCh	146
ckseval1:	57	ClrDA3OK	143
cksneval2:	58	ClrDA3TempF	144
cksseval2:	58	ClrDA3ValidF	145
CKSYMBN	57	ClrDAsOK	144
CKTIME	283	ClrDoStdKeys	137
CKWHEREARGS	67	ClrDouseAlm	283
CLCD10	147	CLRFRC	283
Clean\$	260	ClrLeftAnn	147
Clean\$R0	260	ClrLock	148
CLEARLCD	147	ClrNApKeyOK	137
CLEARMENU	131	ClrNewEditL	139
CLEARVDISP	147	ClrNoRollDA2	146
CLKADJ	107	CLRNUM	112
CLKADJ*	283	ClrNUsrKeyOK	127
ClkOnNib	277	ClrPrgmEntry	148
clkspd	266	ClrRightAnn	147
CLKTICKS	107	ClrServMode	116
CLKUTL1	283	ClrSysFlag	110
CloseUart	116	clrtimeout	283
CLOSEUART	115	ClrUserFlag	110
Clr16	147	ClrUserKeys	127
Clr8	147	CMDSTO	141
Clr8-15	147	CMOS	277
ClrAbbrevStk	283	cmp	5
ClrAlgEntry	148	CMPOBOB	11
ClrAlphaAnn	147	CodePl>%rc.p	123
ClrAppError	137	COERCE	13
ClrAppMode	137	COERCE\$22	33
ClrAppSuspOK	137	COERCE{ }2	13
ClrBusyAnn	148	COERCE2	13
ClrDA1Bad	145	COERCEDUP	13
ClrDA1IsStat	142	COERCEFLAG	87
ClrDA1NoCh	145	COERCESWAP	13
ClrDA1OK	143	COLA	99
ClrDA1TempF	144	COLA_EVAL	99

COLAcase	90	CopyRowsUp	283
COLACOLA	99	COPYVAR	283
COLAITE	89	corner	283
COLAkeep1st	54	count*	70
COLANOTcase	91	count+	70
COLARPITE	89	COVERsave	277
COLASKIP	100	COVERstate	277
COLAskipcola	100	Cr	56
COLAthexFCN	283	CR_COUNT	275
COLATRUE	88	CRC	261
COLCOUNT	277	CREATE	103
colct	63	CREATEDIR	104
Coldstart	283	CREATERRP	103
coleval	64	CREATETEMP	259
colfac	63	CRER	159
colinv1	70	CRLF\$	30
colinv2	70	CROSS_HAIRS	159
colinv3	70	CROSS_OFF	159
colrev	63	CROSSGROB	152
ColumnMAX	45	CROSSMARKON	159
ColumnMEAN	45	CRUNCH	57
ColumnMIN	45	CRUNCHNoBlame	283
ColumnTDEV	45	CSLW5	266
ColumnTOT	45	CSPEED	277
ColumnVAR	45	CSRW5	266
colunfac	64	CtlAlarm	277
COLWIDTH	277	CtlAlarm!	283
COMMANDMASK	272	CtlAlarm@	283
COMPCONFCRC	283	CUREQ	80
COMPEVAL	97	CURRENTMARK?	283
COMPILEID	73	CURRENTMENU	277
COMPN	50	CURSOR	275
ComVecStatus	283	Cursor&Disp	283
Con*Meta	62	CURSOR@	137
Con-Meta	62	CURSOR_END?	137
Con/Meta	62	CURSOR_OFF	138
Con+Meta	62	CURSOR_OFF+	138
Con^Meta	62	CURSOR_OFFO	138
CONFRAM	277	CURSOR+	283
CONFTAB	277	CURSOR1	152
ConmodMeta	62	CURSOR2	152
Connecting	12	cursorblink-	114
constuniterr	256	cursorblink+	114
CONSTUNITERR	86	CURSORCHR	275
CONTEXT	277	CURSOREPOSN	275
CONTEXT!	105	CURSORGROB	275
CONTEXT@	105	CURSOROFFSET	275
CONTRAST	261	CURSORPART	275
convertbase	283	CURSORPOSN	275
CopyColsLeft	283	CURSORROW	275
CopyColsRight	283	CURSORSTATE	275
CopyRowsDown	283	CURSORSX	275

CURSORY	275	DA1NoChMASK	273
D		DA1OK?	144
D/D*	66	DA1OK?NOTIT	144
D/D-	66	DA1TempF	144
D/D/	66	DA1TempMASK	273
D/D=	66	DA1ValidF?	145
D/D+	66	DA1ValidMASK	273
D/D^	67	DA2aBad?	145
D/D^X	67	DA2aBadMASK	273
D/D^Y	67	DA2aLess1OK?	144
D/DABS	66	DA2aNoCh?	146
D/DACOS	66	DA2aNoChMASK	273
D/DACOSH	66	DA2aOK?	144
D/DALOG	66	DA2aOK?NOTIT	144
D/DAPPLY	66	DA2aTempF?	144
D/DARG	66	DA2aTempMASK	272
D/DASIN	66	DA2aValdMASK	273
D/DASINH	66	DA2aValidF?	145
D/DATAN	66	DA2bBad?	145
D/DATANH	66	DA2bBadMASK	273
D/DCHS	66	DA2bEdit?	146
D/DCONJ	66	DA2bIsEdL?	146
D/DCOS	66	DA2bIsEdMASK	272
D/DCOSH	67	DA2bNoCh?	146
D/DDER	67	DA2bNoChMASK	273
D/DEXP	67	DA2bOK?	144
D/DIFTE	67	DA2bOK?NOTIT	144
D/DINTEGRAL	67	DA2bTemp?	284
D/DINV	67	DA2bTempF?	144
D/DLN	67	DA2bTempMASK	272
D/DLNP1	67	DA2bValdMASK	273
D/DLOG	67	DA2bValidF?	145
D/DSIN	67	DA2OK?	144
D/DSINH	67	DA3Bad?	145
D/DSQ	67	DA3BadMASK	273
D/DSQRT	67	DA3NoCh?	146
D/DSUM	67	DA3NoChMASK	273
D/DTAN	67	DA3OK?	144
D/DTANH	67	DA3OK?NOTIT	144
D/DWHERE	67	DA3TempF?	144
D0->Row1	262	DA3TempMASK	272
D0->Sft1	262	DA3ValidF?	145
D0=ALoop	284	DA3ValidMASK	273
D0=DSKTOP	256	DaDGNTc	284
D1=DSKTOP	256	dARRYcase	95
DA1Bad?	145	DAsBad?	145
DA1BadMASK	273	DAsOK?	144
DA1IsStat?	146	DATE	107
DA1IsStatus?	146	DATE+DAYS	107
DA1NoCh?	146	Date+Time	108
		Date>d\$	108
		Date>hxs13	108

Date>wd\$	108	disprange	284
DAY#	284	DISPROW1	150
Day>Date	284	DISPROW1*	150
DCHXW	257	DISPROW1*!	150
DcompWidth	277	DISPROW2	151
DcompWidth@	36	DISPROW2*	151
DCursor	284	DISPROW3	151
DDAYS	107	DISPROW4	151
Debounce	265	DISPROW5	151
DECODE	117	DISPROW6	151
DECOMP\$	37	DISPROW7	151
DeepSleep	266	DISPROW8	151
DEEPSLEEP	112	DispStatus	143
DEG?	111	DISPSTATUS2	151
DELALARM	109	DispStsBound	143
DelayCt	277	DISPTEST	261
delimcase	284	DispTime?	143
DelKey	126	DispTimeMASK	273
DEPTH	77	DispTimeReq?	143
DEPTHSAVE	277	DispVarsUtil	284
derprod1	284	DIV2	284
derquot	284	DIV5	257
dIDNTNcase	95	DIVF	258
DIMLIMITS	41	dLISTcase	95
DIRARGERR	86	DNOTBAKcase	95
DirLabel:	155	DNOTLIBcase	95
DirLabelGrob	152	DNOTSYMB?SEMI	95
DISABLE_KBD	277	DO	100
DisableIntr	256	DO#EXIT	84
DISP@01	150	DO\$EXIT	84
DISP@09	151	DO%EXIT	84
DISP@17	151	DO>LCD	154
DISP@25	151	DO>STR	37
DISP1CTLg	277	Do1st/1st+	148
DISP2CTLg	277	Do1st/2nd+	148
Disp5x7	151	Do1UserMASK	272
DISPADDR	261	DOACPTR	269
DispBadToken	36	DOADJ	106
DISPCHAR+PC	284	DOADJ1	106
DispCoord1	151	DOAPWL	113
DISPCOORD2	151	DOARRY	266
DispDir?Tim1	143	DOARRY>	44
DispDir?Tim2	143	DOARRYPRG1	43
DispDir?Time	143	DOARRYPRG2	43
DispEditLine	143	DoBadKey	126
DispILPrompt	143	DOBAK	266
DISPIO	261	DOBAUD	115
DispMenu	130, 143	DOBEEP	112
DispMenu.1	130, 143	DOBIN	111
DISPN	151	DOBIND	80
DispOff	262	DOBINT	267
DispOn	262	DOBUFLEN	115

DOC>PX	160	DOLIST	268
DoCAAlarmKey	284	DOLNKARRY	268
DOCHAR	267	DOLPENV	277
DOCHR	32	DOMEMERR	256
DOCKSM	115	DoMenuKey	130
DOCLLCD	147	DoMenuKeyNS	131
DOCMP	267	DoMsgBox (~DoMsgBox)	152
DOCODE	267	DoNameKeyLRS	131
DOCOL	267	DoNameKeyRS	131
docola	266	DoNewMatrix	141
docr	116	DoNextRow	132
DOCR	115	DONUM	35
DoCRC	260	DOOCT	111
DoCRCC	260	DoOldMatrix	141
DoCreateMenu	274	DOOPENIO	115
DOCSTR	267	DOPARITY	115
DODEC	111	DOPKT	115
DODELAY	115	DoPlotMenu	284
DoDelim	138	DoPrevRow	132
DoDelims	138	DOPRLCD	284
DODIRPRG	104	DOPX>C	160
DODISP	150	DORANDOMIZE	22
DOECMP	267	DORCLE	159
DOENG	111	DOREAL	268
DOERASE	142	DORECN	114
DOEREL	267	DOROMP	268
DOEXT	267	DOROMPOLL	74
DOEXT0	269	DORRP	268
DOEXT1	269	DOSBRK	115
DOEXT2	269	DOSCI	111
DOEXT3	269	DOSERVER	114
DOEXT4	269	DOSIZEERR	256
DOFINISH	115	doskip	266
DoFirstRow	131	DoSolvvrMenu	131
DOFIX	111	DOSRECV	115
DOGARBAGE	258	DOSTD	111
DOGRAPHIC	159	DoStdKeyMASK	273
DOGROB	267	DoStdKeys?	137
DoHere:	103	DOSTIME	115
DOHEX	111	DOSTOALLFO	110
DOHSTR	267	DOSTOE	159
DOHXS	267	DOSTOSYSF	110
DOHXSEXIT	84	DOSTR>	35
DOIDNT	268	dstws	38
DoInputForm	132	DOSYMB	269
DOKERRM	115	DOTAG	269
DoKeyOb	124	DOTRANSIO	115
DoLabel	131, 155	DOTVARS%	104
DoLam	81	DOUSEALARM	277
DOLAM	268	DOVARS	104
DOLCD>	154	dowait	107
DOLIB	268	dowait/quit?	107

DoWarning	152	DTYPECSTR?	122
dowutil	284	DTYPELIST?	122
DOXMIT	115	DTYPEREAL?	122
DPRADIX?	112	DUMP	81
DRAWBOX#	154	dup	52
DRAWLINE#3	154	DUP	76
drax	284	DUP#<7	16
dREALNcase	95	DUP#0=	16
DREND	277	DUP#0=case	92
DROP	76	DUP#0=csDROP	92
DROP#1-	15	DUP#0=csedrp	92
DROP%0	19	DUP#0=IT	91
DROP%0ABND	19	DUP#0=ITE	91
DROP%1	19	DUP#0_DO	101
DROP%1ABND	19	DUP#0<>	16
DROP'	98	DUP#0<>WHILE	100
DROP2dropf	56	DUP#1-	15
DROP3PICK	79	DUP#1=	16
dropaddoper	54	DUP#1+	15
DropBadKey	126	DUP#1+PICK	53, 76
DROPDEADTRUE	98	DUP#2+	15
dropDROPf	56	DUP#2+PICK	76
DROPDUP	76	DUP\$>ID	80
DROPDUPLEN\$1+	33	DUP%0=	23
DROPFALSE	87	DUP%OCON	42
DropLoop	262	DUP'	98
DROPLoop	101	DUP@	102
DROPNDROP	52, 76	DUP1LAMBIND	81
DROPNULL\$	32	DUP3PICK	76
DROPONE	13	DUP3PICK#+	15
DROPOVER	76	DUP4PUTLAM	83
DROPRDROP	97	DUP4UNROLL	76
DropRepl0	56	DupAndThen	284
DROPROT	76	DUPDUP	76
DROPSWAP	76	DUPEQ:	88
DROPSWAPDROP	76, 77	DUPEXitAtLOOP	101
DROPSYM	57	DUPGROBDIM	153
DropSysErr\$	284	DUPINCOMP	51
DropSysObs	284	DUPINDEX@	101
DROPTRUE	87	DUPLEN\$	33
DROPZERO	13	DUPLENCOMP	48
DRPEXitAtLOOP	102	DUPNULL\$?	37
drpmeta*	69	DUPNULL{ }?	51
drpmeta-	69	DUPNULLCOMP?	48
drpmeta/	69	DUPONE	13
drpmeta+	69	DUPPICK	76
drppargsym	69	DUPROLL	76
drppargtop&	69	DUPROM-WORD?	73
DRSTART	277	DUPROMPTR@	72
DSKTOP	270	DUPROT	76
DTYPEARRY?	122	DUPSAFE@	102
DTYPECOL?	123	DUPTMPENV	83

DUPTEMPEVN	83	EIGHTYONE	8
DUPTWO	13	ELEMENT	277
DUPTYPEARRY?	122	ELEVEN	5
DUPTYPEBINT?	123	Embedded?	49
DUPTYPECHAR?	123	EMPTYDIRERR	86
DUPTYPECMP?	122	ENCODE	117
DUPTYPECOL?	123	ENCODE1PKT	117
DUPTYPECSTR?	122	ENTERCODE	284
DUPTYPEEXT?	122	ENTRWISE	277
DUPTYPEGROB?	122	EQ	88
DUPTYPEHSTR?	122	EQ:	88
DUPTYPEIDNT?	122	EQcase	94
DUPTYPELAM?	122	EQcasedrop	94
DUPTYPELIST?	122	EQCURSOR?	284
DUPTYPEREAL?	122	EQIT	94
DUPTYPEROMP?	122	EQITE	94
DUPTYPERRP?	123	EqList?	51
DUPTYPESYMB?	122	EQLookup	50
DUPTYPETAG?	122	EQOR	88
DUPUNROT	76, 77	EQOVER	88
DUPZERO	13	EqPtr	277
dvarlsBIND	81	EQUAL	88
dvars?	55	EQUALcase	94
dvars>	56	EQUALcasedro	94
DZP	284	EQUALcasedrop	95
		EQUALcasedrp	94
E		EQUALNOT	88
easyabs	284	EQUALNOTcase	94
Echo\$Key	138	EQUALOR	88
Echo\$NoChr00	138	EQUALPOSCOMP	49
Echo2Macros	284	EQUATION	159
EchoChrKey	138	Err#Cont	11
ECUSER	284	Err#Kill	10
ederr	87	Err#NoLstArg	11
EDITDECOMP\$	37	Err#NoLstStk	10
editdecomp\$w	37	ERR0	84
EditExstCase	96	ERRBEEP	83
EDITFLAG	273	ErrFixEIRU	284
EditLevel1	139	Errjmp	256
EditLExists?	137	ERRJMP	84
EDITLFLAG	273	ErrjmpC	256
EDITLINE	275	ERRM	84
EDITLINE\$	137	ERRN	84
EDITLMASK	273	ERRN>HXS	84
EditMenu	140	ERROR	277
EDITPARTS	284	ERROR@	84
EditString	139	ERRORCLR	84
EIGHT	5	ERROROUT	84
EIGHTEEN	6	ERRORSTO	84
EIGHTROLL	78	ERRSET	84
EIGHTY	8	ErrTime	284
		ERRTRAP	84

eval	59	FIFTYNINE	8
EVAL	97	FIFTYONE	7
evalcase:	72	FIFTYSEVEN	8
EVALCRUNCH	284	FIFTYSIX	7
EvalNoCK	121	FIFTYTHREE	7
EvalNoCK:	121	FIFTYTWO	7
evalTRUE	59	Find1stT.1	49
EXAB0	258	Find1stTrue	49
EXAB2	258	FINDALARM%	109
EXCHINITPK	115	FINDALARMLS	109
ExitAtLOOP	101	FINDELN	41
ExitAtLOOPDUP	102	FindNext	109
ExitFcn	284	FINDVAR	57
EXITFCNsto	284	FIRST@LAM	81
EXITMSG	277	FIRSTC@	138
EXITMSGSTO	84	FIRSTC+	138
expan	64	FIRSTCHAR	277
expan1	64	FIRSTPROC	277
EXPAND	34, 38	FIVE	5
expansq	64	FIVEFOUR	9
expansum^	64	FIVEROLL	78
EXPLODE	56	FIVESIX	9
EXPR>	57	FIVETHREE	8
EXT	5	FIVEUNROLL	78
EXTN	45, 50	FixEIRU	284
Extobcode	284	FixRRP	284
EXTOBOB	12	FlashMsg	151
EXTREAL	10	FlashROMPTAB	276
EXTSYM	10	FlashWarning	151
F			
F%>L%	26	FLOAT	284
failed	87	FLOATSYM==	61
FAILSTK1	275	FLOATSYM<>	62
FAILSTK2	275	Flush	265
FAILSTK3	275	FLUSH	124
FAILSTK4	275	FlushAttn	265
FailTime	278	FLUSHKEYS	124
FALSE	87	FLUSHRSBUF	117
FALSE'	98	FNDALARM{}	109
FalseFalse	87	FONTCOUNT	278
FalseTrue	87	FONTHEIGHT	278
FALSETRUE	87	FONTWIDTH	278
FBox/StdLbl:	155	forceadd	65
FcnUtilEnd	284	forcemul	65
FifoByteCt	278	forcemul?aga	65
FIFTEEN	5	FORTY	7
FIFTY	7	FORTYEIGHT	7
FIFTYEIGHT	8	FORTYFIVE	7
FIFTYFIVE	7	FORTYFOUR	7
FIFTYFOUR	7	FORTYNINE	7
		FORTYONE	7
		FORTYSEVEN	7
		FORTYSIX	7

FORTYTHREE	7	GETLINK	74
FORTYTWO	7	GetMat/Vec	285
FOUR	5	getmatchtok	36
FOURFIVE	8	GetMenu%	130
FOURROLL	77	GETMSG	74
FOURROLLROT	77	GETNAME	115
FOURTEEN	5	GetNextToken	36
FOURTHREE	8	getnibs	113
FOURTWO	8	GETPARAM	157
FOURTY	7	GETPIX	285
FOURUNROLL	78	GETPIX3	285
FREEINTEMP?	106	GETPMIN&MAX	158
FStd/BoxLbl:	155	GETPROC	127
FSTMACROROM#	11	GETPTR	256
G			
GARBAGE	113	GETPTRFALSE	263
GARBAGECOL	259	GETPTRLOOP	256
GARBSCRATCH1	278	GETPTRTRUE	263
GARBSCRATCH2	278	GETPTYPE	158
GBUFF	141	GetRes	159
GBUFFGROBDIM	142	GETRES	158
GCOLCOUNT	278	GETRHS	285
GDISP	270	GETRRP	261
GDISPCENTER	159	GETSCALE	159
GDISPON?	141	GETSERIAL	117
get1	53	GetStrLen	266
GETAB0	285	GetStrLenC	266
GETAB1	285	GetStrLenStk	266
GETATELN	41	GETTEMP	260
getBPOFF	285	GETTHEMSG	85
GetBVars	285	GetTimChk	285
GETCDO	285	GetTime++	285
GetChkPRTPAR	116	GETTOUCH	124
GETCONFIG	74	GetUserKeys	126
GETDF	127	getwordsize	266
GETEL	41	GETXMAX	157
GetElt	285	GETXMIN	157
GetEqN	159	getxpos	159
GETEXITMSG	84	GETXPOS	159
GETHASH	74	GETYMAX	158
GETINDEP	158	GETYMIN	157
GetIOPAR	116	getypos	159
GetKermPkt#	116	GETYPOS	159
GETKEY	124	gFldVal (~gFldVal)	285
GETKEY*	124	GOTO	97
GetKeyOb	124	GPErrjmpC	256
GETKP	117	GPMEMERR	256
GETLAM	81	GPOverWrALp	263
GETLAMPAIR	83	GPOverWrFLp	263
GetLastEdit	285	GPOverWrROLp	263
		GPOverWrT/FL	263
		GPOverWrTlp	263
		GPPushA	263

GPPushFLoop 263
 GPPushFTLp 263
 GPPushT/FLp 263
 GPPushTLoop 263
 GraphicExit 285
 GraphPrtHook 278
 grob 5
 grob! 262
 GROB! 153
 GROB!ZERO 153
 GROB!ZERODRP 153
 GROB+# 153
 GROB>GDISP 153
 Grob>Menu 131, 155
 grobAlertIcon (~grobAlertIcon) 152
 grobCheckKey (~grobCheckKey) 152
 GROBDIM 152
 GROBDIMw 153
 GsstFIN 285

H

H/W>KeyCode 123
 H/wKey>KeyOb 123
 HANDSHK 277
 HARDBUFF 141
 HARDBUFF2 141
 HARDHEIGHT 142
 HARDRAMEND 278
 HARDROMEND 277
 HBUF_X_Y 142
 HBUFF_X_Y 149
 HBUFFDIMw 142
 HEIGHTENGROB 142
 HEIGHTENHBUFF 142
 HISTON? 285
 HISTORY1 278
 HISTORY2 278
 HISTORY3 278
 HISTORY4 278
 HISTORYLEVEL 278
 HOMEDIR 105
 HOMEMASK 278
 HRAMEND 278
 HSCALE 159
 HXDCW 257
 hxs 5
 HXS#HXS 40
 HXS==HXS 40
 HXS># 13
 hxs>\$ 32
 HXS>\$ 32

HXS>% 19
 HXS>=HXS 40
 HXS>HXS 40
 HXS<=HXS 40
 HXS<HXS 40
 HXSREAL 10

I

ICMPDRPRTDRP 51
 id 5
 ID_CST 80, 131
 ID_S 80
 ID_SIGMADAT 80
 ID_SKEY 80
 ID_UserKeys 80
 ID_UserKeys 80
 ID_X 80
 ID_Y 80
 ID>\$ 32
 ID>LAM 80
 Id>Menu 131, 155
 ID>TAG 40
 IDCONFERR 86
 IDLISTOB 11
 idnt 5
 idntcase 95
 idntlamcase 95
 IDREAL 9
 IDREALOB 11
 IDUP 97, 100
 IFMenuRow1 (~IFMenuRow1) 132
 IFMenuRow2 (~IFMenuRow2) 132
 IgnorAlmMASK 272
 ImmedEntry? 148
 IMPLODE 56
 INCOMPDRP 51
 IncrLAMPKNO 116
 INDEPVAR 158
 INDEX@ 101
 INDEX@#- 101
 INDEX@- 101
 INDEXSTO 101
 infarg? 52
 infreserr 257
 INFRESERR 86
 INHARDROM? 113
 InitDispModes 146
 InitEd&Modes 141
 InitEdLine 141
 InitEdModes 141
 INITEN 278

InitEnab	285	IRAMHOMEmsn	276
InitIOEnv	115	IRAMMASK	278
InitMenu	130	IRC	285
InitMenu%	130	IREG	278
InitOldMat	285	IStackKey	285
InitPOLVars	137	ISTOP-INDEX	101
InitSysUI	285	ISTOP@	101
InitTrack:	129	ISTOPSTO	101
INNER#1=	51	IT	89
INNERCOMP	51	ITE	89
INNERDUP	51	ITE_DROP	89
INNERtop&	51	ITEM1LINES	278
InputLAttn	132	ITEM1STATE	278
InputLEnter	132		
InputLine	132	J	
INPUTSTREAM	278	j%0=case	92
INSERT?	139	JEQcase	94
INSERT_MODE	138	JINDEX@	101
InsertEcho	138	JINDEXSTO	101
INSERTMASK	272	JstGetTHEMESG	84
INSERTN{}	51	JstGETTHEMSG	84
InSimplyExpr	274	JSTOP@	101
IntDiv	257	JSTOPSTO	101
INTEGER337	11	JUMPBOT	150
INTEMNOTREF?	106	JUMPLEFT	150
INTEMPOB? (~INTEMPOB?)	106	JUMPRIGHT	150
intg	65	JUMPTOP	150
intg1	65		
intg1fail	65	K	
intg1ok	65	KDispRow2	115
intgaddlin	65	KDispStatus2	115
intgconst	65	KeepUnit	46
intglinear	65	KERMERM	278
INTRAM	271	KERMMODE	278
INTRPPTR	270	KERMOOPEN	115
intrptderr	257	kempktmsg	117
InvalServCmd	12	kermrecvmsg	118
INVDEFERR	86	kerm sendmsg	118
INVFUNCERR	85	Key>StdKeyOb	127
INVGROB	153	Key>U/SKeyOb	127
InvLabelGrob	152	KEYBUFFER	278
INVUNITERR	86	KeyInBuff?	265
IOC	285	KEYINBUFFER?	125
IOCheckReal	116	KEYLIST	278
IOCNIB	278	KEYLOCK	279
IOCsave	278	KeyOb	279
IOSAVE	278	KeyOb!	123
IRAM@	285	KeyOb@	123
IRAMBEND	278	KeyOb0	123
IRAMBSIZE	278	KeyRomPtr0	279
IRAMBUFF	278		
IRAMBUFF2	278		

KeyRomPtr1	279	LASTCMDERR	86
KeyRomPtr2	279	LastContext	279
KeyRomPtr3	279	LastERow?	285
KeyRomPtr4	279	LASTERROR	279
KeyRomPtr5	279	LASTLAST?	279
KeyRomPtr6	279	LastMenuDef	279
KEYSTATE	279	LastMenuDef!	128
KILLADISP	142	LastMenuDef@	128
KILLGDISP	142	LastMenuRow	279
KINVISLF	117	LastMenuRow!	128
KVIS	117	LastMenuRow@	128
KVISLF	116	LastNonNull	104
L			
L%*	26	LASTOP	279
L%-	26	LastPrntTime	279
L%/	26	LASTPT?	285
L%+	26	LASTRAM-WORD	104
LabelDef	279	LASTROMWDOB	279
LabelDef!	127	LASTSTKERR	85
lam	5	lbrac	285
lam'dvar	80	LCursor	285
LAM_SKEY	80	leeway	281
LAM>ID	80	LEFTCOL	149
LAMERRMSG	114	LEFTTREE	279
LAMEXCHP	114	LEN\$	32
LAMKLIST	114	LEN\$>%	32
LAMKML	114	LENCOMP	48
LAMKMODE	114	LENCOMP>%	48
LAMKP	114	LENHXS	38
LAMKPTRN	114	LIB>#	74
LAMKRM	114	linear	66
LAMLNAME	114	linear!	66
LAMMaxR	114	linear?	66
LAMOBJ	114	LineByteCt	279
LAMOPOS	114	LINECHANGE	285
LAMPACKET	114	LINECOUNT	261
LAMPKNO	114	LINECOUNTg	279
LAMRETRY	114	LineGrob	152
LAMSavedUI	136	LINENIBSg	279
larg	69	LINEOFF	154
larg&	69	LINEOFF3	154
LAST\$	34	LINEOFFS	261
LASTARG	279	LINEON	154
LASTARG1	279	LINEON3	154
LASTARG2	279	LINESOFSTACK	285
LASTARG3	279	list	5
LASTARG4	279	List	285
LASTARG5	279	LISTCMP	8
LASTARGCOUNT	279	LISTLAM	9
LASTARGf	279	LISTLISTOB	11
		LISTRCL	102
		LISTREAL	8
		LISTREALOB	11

LISTREALREAL	11	MANMENUCX	286
liteslp	266	MANMENU EQ	286
LiteSlp	113	MANMENU EXP	286
LoadTouchTbl	130	MANMENU LN	286
LoBatTime	279	MANMENU TRG	286
Lock?	148	MARKGROB	152
LockAlpha	148	MAT*	42
LOCUPSIZE	276	MAT-	42
longhxs	285	MAT/	42
Lookup	49	MAT+	42
Lookup.1	50	MATABS	43
Loop	285	MATC>R	43
LOOP	101	matchob?	49
lowbaterr	257	matchob?Lp	286
LOWERMASK	272	MATCNRM	43
LPD_HIST	279	MATCON	41
		MATCONJ	42
		MATCROSS	43
		MATDET	43
		MATDOT	43
		MATFLOAT*	42
		MATFLOAT/	42
		MATIDN	41
		MATIM	43
		MATINV	42
		MATNEG	42
		MATR>C	43
		MATRE	43
		MATREDIM	42
		MATRND	43
		MATRNRM	43
		MATRSD	43
		MATSQ	42
		MATTRN	42
		MATTRNC	43
		MAXRETRY	114
		MBox/StdLbl:	155
		MDIMS	41
		MDIMSDROP	41
		mEditLExists	276
		MediumCursor	152
		MEM	113
		MEMERR	5
		MEMSKIP	100
		MENoP&FixDA1	146
		MENP&FixDA12	146
		MENUADDR	261
		MenuData	279
		MenuDef	279
		MenuDef@	128
		MenuExitAct	279
		MenuExitAct!	128
M-1potcase	94		
M-1stcasechs	94		
M1st*case	93		
M1st-case	93		
M1st/case	93		
M1st+?Drp	68		
M1st+case	93		
M1st^case	94		
M1stFNCcase	94		
M1stINVcase	94		
M1stNEGcase	94		
M1stSQcase	94		
MACRODCMP	285		
MAKE\$	260		
MAKE\$N	260		
MAKEARRY	44		
makebeep	266		
MakeBoxLabel	154		
MakeDirLabel	154		
makegrob	262		
MAKEGROB	153		
MAKEHXS	38		
MakeInvLabel	155		
MakeLabel	131, 156		
MAKEPICT#	153		
MAKEPVARs	157		
MAKERRP	103		
MakeStdLabel	154		
MANMENU*/	285		
MANMENU+-	285		
MANMENU^	286		
MANMENUATG	285		
MANMENUCSIV	285		

MenuKey	131	Meta<-Aall	71
MenuKeyLS	280	Meta<-D	71
MenuKeyLS!	128	Meta<-D!	71
MenuKeyNS	280	Meta<-Dall	71
MenuKeyNS!	129	Meta<-M	71
MenuKeyNS@	129	Meta<-Mall	71
MenuKeyRS	280	Meta<-T	70
MenuKeyRS!	129	Meta<-Tall	71
MENULEVEL	279	Meta1/	70
MenuMaker	130	Meta1/()	71
MENUOFF	142	MetaA->	70
MENUOFF?	142	MetaA->all	71
MenuRow	280	MetaABS	62
MenuRow!	128	MetaAF	71
MenuRow@	128	MetaALOG	63
MenuRowAct	280	MetaConcase	93
MenuRowAct!	129	MetaCONJ	62
MEQ1stcase	93	MetaCOS	63
MEQopscase	93	MetaCOSH	63
MEQU?	55	MetaD->	71
MERGE	72	MetaD->all	71
MESRclEqn (~MESRclEqn)	286	MetaDINV	70
Meta()	70	MetaDNEG	70
Meta(<-	70	MetaE()	71
Meta(<-all	71	MetaE^	71
meta*	69	MetaEXP	63
Meta*	62	MetaEXPM	63
Meta*1	70	MetaIFTE	68
Meta*Con	62	MetaIM	62
meta-	69	metainv	69
Meta-	62	MetaINV	62
Meta-()	71	metainvlft	69
Meta->()	71	MetaL()	71
Meta->()all	72	MetaL*	71
Meta->()C%	71	MetaLess?	64
Meta->()C%all	72	MetaM->	71
Meta->)	70	MetaM->all	71
Meta->)all	72	Metamod	62
Meta->DEF	71	MetamodCon	62
Meta->TRG	71	MetaMore?	64
Meta-Con	62	MetaMulInv	70
meta/	69	metaneg	69
Meta/	62	MetaNEG	62
Meta/Con	62	metaneglft	69
meta+	68	MetaRCOLCT	70
Meta+	62	MetaRE	62
Meta+1-1	70	metaROTDUP	53
Meta+Con	62	MetaSIN	63
Meta^1	70	MetaSINH	63
Meta^Con	62	MetaSQ	63
Meta<->	70	MetaT->	70
Meta<-A	70	MetaT->all	71

metatail.....	55	NEXTLIBBAK.....	72
MetaTAN.....	63	nextpos.....	286
MetaTANH.....	63	NEXTROMPID.....	74
MetaTRG*.....	71	NEXTRRPOB.....	286
MetaUnCalc.....	54	NEXTSTEP.....	286
METREDIM.....	41	nextsym'R.....	286
Mid1stcase.....	93	ngsizecase.....	286
MINUSONE.....	12	NINE.....	5
MISLIBERR.....	86	NINETEEN.....	6
Modifier.....	130	nINTGACOS.....	286
ModifierKey?.....	123	nINTGALOG.....	286
MOVEDOWN.....	259	nINTGASIN.....	286
MOVEDSD.....	259	nINTGATAN.....	286
MOVEDSU.....	259	nINTGCOS.....	286
MOVERSD.....	259	nINTGCOSH.....	286
MOVERSU.....	259	nINTGEXPM.....	286
MOVEUP.....	259	nINTGINV.....	286
MOVEVAR.....	103	nINTGLN.....	286
mpop1%.....	286	nINTGLOG.....	286
MPY.....	257	nINTGSIGN.....	286
MsgBoxMenu (~MsgBoxMenu).....	152	nINTGSIN.....	286
MUL#.....	257	nINTGSINH.....	286
MULTF.....	258	nINTGSQ.....	286
MYRAMROMPAIR.....	104	nINTGSQRT.....	286
N			
N+1DROP.....	52, 76	nINTGTAN.....	286
N+1roll.....	53	nINTGTANH.....	286
N+1unroll.....	53	nmetasyms.....	121
NAppKeyMASK.....	273	nNullBind (~nNullBind).....	81
NAppKeyOK?.....	137	NOALARMERR.....	87
NcaseSIZEERR.....	96	NOALARMSRV.....	280
NcaseTYPEERR.....	96	NoAlgProcess.....	274
nCOLCTQUOTE.....	286	NoAppDisplay!.....	136
nCustomMenu.....	131	NoAppError!.....	137
NDROP.....	52, 76	NoAppExitCnd!.....	137
NDROPFALSE.....	52, 87	NoAppKeys!.....	136
NDUP.....	76	NoAttn?Semi.....	125
NDUPN.....	76	NOBLINK.....	114
need'case.....	286	NODECOUNT.....	280
NEGFLOWERR.....	86	NoEdit?case.....	96
negunferr.....	257	NoEditLine?.....	137
newBASE.....	286	NOEQERR.....	85
NewEditLMASK.....	273	NoExitAction.....	128
NEWINDEP.....	286	nohalt.....	137
NEWLINE\$.....	30	NOHALTERR.....	86
NEWLINE\$\$.....	33	NoIdsInMeta?.....	55
NEWLINE&\$.....	33	NoIgnoreAlm.....	286
NEWMARK.....	286	nonopcase.....	95
NEXTCOMPOB.....	50	NonUsrKeyOK?.....	127
NEXTIRQ.....	280	NOP.....	96
		NOP1MASK13.....	274
		NOP1MASK14.....	274
		NOP1MASK15.....	274

NOP1MASK16	274	NOTcaseFALSE	90
NOP1MASK17	274	NOTcaseTRUE	90
NOP1MASK18	274	NOTCONST?	112
NOP1MASK19	274	NOTcsdrpfls	90
NOP1MASK20	275	NOTINHARDROM?	113
NOP2MASK12	273	NOTLISTcase	95
NOP2MASK13	274	NOTNUM?	112
NOP2MASK14	274	NOTREF?	106
NOP2MASK15	274	NOTROMPcase	95
NOP2MASK16	274	NOTSECOcase	95
NOP2MASK17	274	nscknum2	58
NOP2MASK18	274	NTH@LAM	81
NOP2MASK19	274	NTHCOMDDUP	48
NOP2MASK20	275	NTHCOMPDROP	48
NOP4MASK12	273	NTHELCOMP	48
NOP4MASK13	274	NTHOF	49
NOP4MASK14	274	NULL\$	29
NOP4MASK15	274	NULL\$?	37
NOP4MASK16	274	NULL\$\$SWAP	32
NOP4MASK17	274	NULL\$TEMP	32
NOP4MASK18	274	NULL::	52
NOP4MASK19	274	NULL{}	51
NOP4MASK20	275	NULLCHARERR	85
NOP8MASK12	273	NULLCOMP?	48
NOP8MASK13	274	NULLHXS	38
NOP8MASK14	274	NULLHXS?	38
NOP8MASK15	274	NULLID	79
NOP8MASK16	274	NULLLAM	80
NOP8MASK17	274	NullMenuKey	129
NOP8MASK18	274	NullMenuLbl	152
NOP8MASK19	274	NULLPAINT	152
NOP8MASK20	275	NULLSYMB	56
norecCSseq	286	nultrior	36
norecPWLseq	289	num-1=case	93
NormAppFlags	136	num0=case	92
NoRolDA2MASK	272	num1=case	92
NoRollDA2?	146	num2=case	93
NOT	88	numb1stcase	95
NOT\$	35	NumbMode	111
NOT?DROP	89	NumIFTE	68
NOT?GOTO	97	NUMINTEG	65
NOT?SEMI	89	NUMSOLVE	287
NOT?SWAPDROP	89	NUsrKeyMASK	273
NOT_IT	89	nWHEREER	287
NOT_UNTIL	100	nWHEREFCNAPP	287
NOT_WHILE	100	nWHEREIFTE	287
NOTAND	88	nWHEREINTG	287
NOTcase	89	nWHEREISUM	287
NOTcase2drop	90	nWHEREWHERE	287
NOTcase2DROP	90	NXTPOT%	64
NOTcasedrop	90		
NOTcaseDROP	90		

O

O*SYM	59
O-SYM	59
O/SYM	59
O+SYM	59
O^SYM	59
OB/EXP?	280
OB>BAKcode	287
Ob>Seco	52
ObEdit	139
ObInMeta?	55
OBTREELEN	280
OBUPSIZE	276
OBUPSTART	270
OCRC	113
OCRC%	113
OFFSRRP	73
ofloerr	257
OLDMENU	280
ONE	5
ONE#>	16
ONE_DO	101
ONE_DO_ARRAY	101
ONE_EQ	16
ONE{ }N	51
ONECOLA	99
ONEDUP	12
ONEFALSE	13
ONEFALSE'	98
ONEHUNDRED	9
ONEONE	12
ONEPOS\$	33
ONESWAP	13
ONESYMBN	50
OnKeyDown?	265
OnKeyStable?	265
ONSRRP?	73
OpenIO	115
OpenIOPrnt	115
OpenUart?Clr	116
OpenUartClr	116
OR	88
OR\$	35
ORcase	90
ORDERXY#	154
ORDERXY%	154
ORghost	280
ORNOT	88
OSAVE	280
OSIZE	113
OUTUART	116
OVER	79
OVER#-	15
OVER#=	16
OVER#=case	91
OVER#+	15
OVER#>	16
OVER#<	16
OVER#0=	16
OVER#2+UNROL	53
OVER#2+UNROLL	79
over&	53
OVER'	98
OVER5PICK	79
OVERARSIZE	41
OVERDUP	79
overev&	53
OVERFLOWERR	86
OVERINDEX@	101
OVERLEN\$	33
OVERSWAP	79
OVERUNROT	79
OverWrF/TLp	287
OverWrFLoop	263
OverWrT/FLp	263
OverWrTLoop	263

P

P::N	50
P{ }N	50
PACK	258
PACKSB	258
PADCOUNT	280
PADJSAVE1	280
PADJSAVE2	280
PAINTTREE	280
palparse	35
PALPTRDCMP	287
palrompdcmp	37
parameval	58
PARENCOUNT	280
ParenModFLAG	273
ParenModmask	276
ParenModMASK	273
PARENTTREE	280
parg&	69
pargop	69
ParOuterLoop	136
Parse.1	36
ParseFail	36
PASTDUE	280
PASTDUE#	280
PATHDIR	104

PathStatus	287	PORT2EOS	280
PCunpack (~PCunpack)	287	PORTEND	72
PDCHXS	280	portnotaverr	257
PDCSYMB	278	PortStat	280
PFIFO	280	PORTSTATUS	72
PgUserKeys	127	POS\$	33
PI/180	19	POS\$REV	33
PICK	79	POSCHR	33
pick1	53	POSCHRREV	33
pick1#0=case	93	POSCOMP	48
PICTRCL	156	POSFLOWERR	86
PIXOFF	153	posunferr	257
PIXOFF3	154	POWERCTRL	262
PIXON	153	POWERSTATUS	261
PIXON?	153	PRECSTACK	280
PIXON?3	154	preFACT	287
pixon2	287	PREMARKON	287
PIXON3	153	PrevNonNull	104
plDRPpZparg	69	PREVRAM-WORD	104
PLOTERR	160	PrgmEntrStat	287
PlotOneMore?	160	PrgmEntry?	148
PLOTPREP	160	PRINT	116
PointDerivUt	287	PrintGrob	116
PointMoveCur	287	PRINTINGMASK	272
POLErrorTrap	287	PRINTxNLF	116
POLKeyErr	136	PRLG	287
POLKeyUI	136	PROGIDCMP	11
POLRestoreUI	136	PROGIDEXT	11
POLResUI&Err	136	PROGIDLIST	11
POLSaveUI	136	PROGIDREAL	11
POLSet&KeyUI	136	PromptIdUtil	34
POLSetUI	136	PROTERR	12
POP#	263	PRSOL?	112
POP1%	264	prtparerr	257
POP1%SPLITA	264	PrtStatus	280
POP2#	263	psh	52
POP2%	264	psh&	53
POP2HXS	264	psh1&	54
PopASavptr	263	psh1&rev	54
POPC%	264	psh1&rev:	54
POPC%/%	264	psh1&rev2:	54
POPDATE%	287	psh1top&	53
popflag	263	pshm1	53
POPHXS	264	pshpullpsh1&	54
POPKEY	265	pshtop&	53
POPPEDKEY	280	pshzer	54
PopSavptr	263	pshzerpsharg	69
POPTIME%	287	PSubErr	287
POPUART	116	PSYMBN	51
PORTOEOS	280	PtoR	287
Port1CRC	280	PTR>ID	73
PORT1EOS	280	PTR>ROMPTR	73

PTRREFD?	106	PUTSERIAL	117
PTYPE>PINFO	287	PutSerialEck	287
PuHiddenVar	105	PUTXMAX	157
pull	15, 53	PUTXMIN	157
PULLCMPEL	42	PUTYMAX	158
pulldrop	53	PUTYMIN	157
pulldroppull	54	PvarsC%0	287
PULLEL	42	pZpargSWAPUn	69
PULLEREALEL	42		
PULLLONGEL	42	R	
pullpsh1&	54	R@	97
pullpshm1	53	R>	97
PULLREALEL	42	R1[A]save	271
pullrev	54	R2[A]save	271
pullrev1&	54	R2[S]save	271
puretemp?	47	RAD?	111
PURGALARM%	109	RADD1	258
PURGE	103	RADDF	258
PUSH#	263	RAM-WORDNAME	104
PUSH#ALOOP	264	RAMEND	280
Push#FLoop	264	RAMSTART	277
Push#Loop	263	RBR	287
PUSH#LOOP	264	RCAB0	257
Push#TLoop	264	RCAB2	258
PUSH%	264	RCCD0	258
PUSH%LOOP	264	RCCD2	258
PUSH2#	264	RCKBp	287
Push2#aLoop	264	Rcl&Do:	140
Push2#Loop	264	RCLALARM%	109
PUSHA	263	RCLALM	109
PUSHC%	264	RclHiddenVar	105
PUSHC%%	264	RclHPUI	136
PushF/TLoop	263	RCLSYSF	110
PushFLoop	263	RclUI	136
PushFTLp	263	RCLUSERF	110
PUSHhxs	264	RCS	287
PUSHhxsLoop	264	RDROP	97
PushT/F	263	RDROPCOLA	97
PushT/FLoop	263	RDROPCOLATRUE	97
PushTLoop	263	RDROPFALSE	88
PUTAB0	287	RDUP	97
PUTCMPEL	42	Re>C%	24
PUTEL	42	real	5
PUTINDEP	158	REALcase	95
PUTINDEPLIST	158	REALEXT	6
PUTLAM	82	REALNEGcase	93
PUTLIST	51	REALOB	5
putnibs	114	REALOB0B	10
PUTPTYPE	158	realPACode	287
PUTREALEL	42	REALREAL	6
PUTRES	158	REALREAL0B	10
PUTSCALE	159		

REALSTRSTR	11	Roll&Do:	140
REALSYM	6	ROLL{}	51
RebuildMASK	272	roll2top&	53
RECLAIMDISP	142	ROLLDROP	78
RECORDX&YC%	287	ROLLSWAP	78
RecvNextPkt	116	rolltwotop&	53
REDISPHBUF	149	Rom-Word?	287
REFERENCED?	106	ROM-WORD?	73
REMAP	116	ROMPANY	10
REMOVEN{}	51	ROMPART	74
REPEAT	100	ROMPART>ADDR	74
REPEATER	124	ROMPARTNAME	74
REPEATERCH	125	ROMPARTS	280
REPKEY?	124	ROMPARTSIZE	74
repl%-1	54	rompointer	5
repl%1	54	ROMPOLL	74
repl*	68	ROMPTAB	276
Repl-1	56	ROMPTR@	72
repl/	68	ROMPTR@NOT	102
repl:	54	ROMPTR>#	72
Repl0	56	ROMPTRDECOMP	73
Repl0ABND	56	ROMSEC	72
Repl1	56	ROOM	259
REPLACE	103	RootUtil	287
REPLACE_MODE	287	rot	52
replfunc	54	ROT	77
REQcase	94	ROT#-	15
REQcasedrop	94	ROT#+	15
ReqClkOnMASK	272	ROT#+SWAP	15
RESETDEPTH	77	ROT#1+	15
RESOROMP	73	ROT#1+UNROT	15
RESRAMEND	276	ROT+SWAP	15
Restore16	151	ROT2DROP	76, 77
restoreiram	287	ROT2DUP	77
RestVarRes	105	ROTAND	88
Retry	12	ROTDROP	77
revalcase:	72	ROTDROPSWAP	77, 78
reversym	77	ROTDUP	77
ReviewKey	280	ROTOVER	77
ReviewKey!	129	ROTROT2DROP	77, 78
revpull&psh	54	rotswap	53
revpulldrop	54	ROTSWAP	77
rGETATELN	41	ROTUntop&	53
RIGHT\$3x6	156	RowElt#	288
RIGHTCOL	149	ROWNUM	288
RIGHTTREE	280	Rows8-15	149
RNDC[B]	287	RPIT	89
RNDSYM	60	RPITE	89
RNDXY	21	rpnXROOT	288
RNSEED	280	RPTRACC	288
rNTHELCOMP	48	RROLL	98
ROLL	78	RSKIP	98

RSKTOP	270	Seco>Menu	131, 155
rstfmt1	111	SEMAPH	280
RSUB1	258	SEMI	98
RSWAP	97	SEMILOOP	101
RSZVDISP	142	SEND_PACKET	117
		SENDACK	117
S		SENDEOT	115
SAFE@	102	SENDEROR	115
SAFE@_HERE	102	SENDLIST	114
SAFE@NOT	102	SENDNAK	115
SAFESKIPOB	288	SENDNAME	114
SAFESTO	103	SENDNULLACK	117
sALLOWINTR	288	SENDPKT	115
SAVE_A	271	SendSetup	116
SAVE_B	271	SEP\$NL	34
SAVE_BO	271	ServModeMASK	272
SAVE_C[A]	271	SET	288
SAVE_D	271	SetAbbrevStk	288
SAVE_DO	271	SetAlgEntry	148
SAVE_LC	271	SetAlphaAnn	147
SAVE_LN	271	SetAppError	137
SAVE_MODES	271	SetAppMode	137
SAVE_OFFSET	271	SetAppSuspOK	137
SAVE_OR	280	SetBadPOLUI	288
SAVE_PC	271	setbeep	112
SAVE_R0	271	SETCIRCERR	86
SAVE_ST	271	SETCORPORT	85
Save16	151	SetCursor	139
SAVECLK	271	SETCURSOR	139
SAVECROSS	271	SetDA123NoCh	146
SavedUILS	136	SetDA12a3NCh	146
SAVEERRN	84	SetDA12a3NoCh	146
savefmt1	111	SetDA12NoCh	146
SAVELAM	288	SetDA12Temp	144
SaveLastEdit	141	SetDA13NoCh	146
SaveLastMenu	128	SetDA1Bad	145
SAVESTACK	81	SetDA1BadT	145
SaveVarRes	105	SetDA1IsStat	146
SAVPTR	256	SetDA1NoCh	145
SavPtrTime*	288	SetDA1Temp	144
sBEG	288	SetDA1TempF	144
sBPOFF	288	SetDA1Valid	144
ScreenDn	150	SetDA1ValidF	145
ScreenDnN	150	SetDA23NoCh	146
ScreenUp	150	SetDA2aBad	145
ScreenUpN	150	SetDA2aBadT	145
SCROLLDOWN	149	SetDA2aEcho	146
SCROLLLEFT	149	SetDA2aNoCh	146
SCROLLRIGHT	149	SetDA2aTemp	144
SCROLLUP	149	SetDA2aTempF	144
seco	5	SetDA2aValid	144
		SetDA2aValidF	145

SetDA2bBad	145	SETOBINUSE	85
SetDA2bBadT	145	SETPORTNOTAV	85
SetDA2bEdit	146	SetPrgmEntry	148
SetDA2bIsEdL	146	SETRAD	111
SetDA2bNoCh	146	SetRebuild	129
SetDA2bTemp	144	SetRightAnn	147
SetDA2bTempF	144	SETROMPART	288
SetDA2bValid	144	SETROMPERR	85
SetDA2bValidF	145	SetServMode	116
SetDA2Echo	146	SETSIZEERR	86
SetDA2NoCh	146	SetSomeRow	131
SetDA2OKTemp	144	SETSTACKERR	86
SetDA2Valid	145	setStdwid	36
SetDA3Bad	145	SetSysFlag	110
SetDA3BadT	145	SetThisRow	130
SetDA3NoCh	146	setTimeout	288
SetDA3Temp	144	SETTYPEERR	86
SetDA3TempF	144	SetUserFlag	110
SetDA3Valid	145	SETXNONEXT	85
SetDA3ValidF	145	SEVEN	5
SetDAsNoCh	146	SEVENROLL	78
SetDAsTemp	144	SEVENTEEN	6
SetDAsValid	145	SEVENTY	8
SETDEG	111	SEVENTYFOUR	8
SETDIRRECUR	85	SEVENTYNINE	8
SetDoStdKeys	137	Sfkey1	288
SetEcma94	116	Sfkey6	288
SETFIRSTC_0	138	sFldVal (~sFldVal)	288
setflag	288	sFLUSH	288
SETGRAD	112	ShowInvRomp	112
SETHASH	74	SHOWLS	68
SetHiddenRes	105	Shrink\$	260
SETINVPPAR	86	Sig?ErrJmp	87
SetIOPARerr	86, 116	SINNER	56
SETISOLERR	86	SINNERMETA	56
SetISysFlag	288	SIX	5
SETIVLERR	86	SIXROLL	78
SetKeysNS	129	SIXTEEN	5
SETLAMERR	85	SIXTY	8
SETLBERR	85	SIXTYEIGHT	8
SetLeftAnn	147	SIXTYFOUR	8
SetLock	147	SIXTYONE	8
SETLOOPENV	288	SIXTYTHREE	8
SETMEMERR	85	SIXTYTWO	8
SETMSG	74	SIXUNROLL	78
SetNAppKeyOK	137	SizeMLDisp	274
SETNONERAL	86	SKIP	99
SETNONEXTERR	86	skipcola	100
SetNoRollDA2	146	SKIPOB	259
SETNOROOM	85	SLEEPxcp	288
SETNUM	112	SLOW	106
SetNUsrKeyOK	127	SmallCursor	152

sncknum2	58	StdIOPAR	116
sNEGATE	288	StdLabelDef	155
SolvMenuInit	131	StdMenuKeyLS	129
SORTASLOW	107	StdMenuKeyNS	129
SPACE\$	30	StdPRTPAR	116
SPLITA	258	Stk0save	271
splitdown	69	Stk1save	271
SPLITEQ	57	Stk2save	271
SPLITmsg	85	Stk3save	271
splitup	69	Stk4save	271
SPLITWHERE	288	Stk5save	271
SPLTAC	258	STKDCMASK	272
SQRF	288	stkdecomp\$w	37
srvc_timer2	288	STO	103
SrvcKbdAB	266	STO'	98
sscknum2	58	STOALARM%	109
sstDISP	150	STOALARMLS	108
STAB0	257	STOALLFcont	110
STAB2	257	STOALM	108
stackitw	288	STOAPPLDATA	288
STACKNUM	281	StoHiddenVar	105
StartMenu	130	StoIOPAR	116
StartTime	281	STOLAM	81
StartupProc	288	STOPLOOP	101
STATCLST	44	StoPRTPAR	116
STATCOL	44	STOPSIGN	281
STATCORR	45	STOPSIGN!	105
STATCOV	45	STOPSIGN@	105
STATGETXCOL	45	STOSYSF	110
STATGETYCOL	45	STOUSERF	110
STATLR	45	StoUserKeypatch	126
STATMEAN	44	StoUserKeys	126
STATN	44	str	5
STATPREDX	45	Str>Menu	131, 155
STATPREDY	45	STRETCHCOUNT	281
STATRCL	44	STRIPTAGS	40
STATSMAX	44	STRIPTAGS12	40
STATSMIN	44	STRLIST	7
STATSTDEV	44	STRREALREAL	11
STATSTO	44	sTRUNC	288
STATTOT	44	SUB\$	34
STATVAR	44	SUB\$1#	34
STATX	45	SUB\$SWAP	34
STATXCOL	44	SUBCOMP	49
STATXX	45	SUBGROB	153
STATXY	45	SUBHXS	38
STATY	45	SubMeta0b	55
STATYCOL	44	SubMeta0b1	55
STATYY	45	subpdcdptch	288
STCD0	257	SuspendOK?	137
STCD2	257	SW_ETime	281
Std/BoxLabel	155	SW_Image	281

SWAP	77	SYM%CH	60
SWAP#-	15	SYM%COMB	60
SWAP#1-	15	SYM%MAX	59
SWAP#1-SWAP	15	SYM%MIN	59
SWAP#1+	15, 53	SYM%MOD	59
SWAP#1+SWAP	15	SYM%OF	60
SWAP%/	22	SYM%OR	61
SWAP%>C%	24	SYM%PERM	60
SWAP&\$	35	SYM%RND	60
SWAP'	98	SYM%SUM	68
SWAP>HCOMP	48	SYM%T	60
SWAP2C%>%%	25	SYM%TRNC	60
SWAP2C%>%	24	SYM%XOR	61
SWAP2DUP	77	SYM%XROOT	60
SWAP3PICK	77	SYM*	59
SWAP4PICK	79	SYM*0	59
SWAP4ROLL	77	SYM-	59
SWAPCKREF	106	SYM-0	59
SWAPCOLA	99	SYM/	59
SWAPCOLUMNS	41	SYM/0	59
SWAPcompSWAP	57	SYM==	61
SWAPDROP	77	SYM+	59
SWAPDROP#1-	15	SYM+0	59
SWAPDROPDUP	77	SYM>	62
SWAPDROPLLOOP	101	SYM>=	62
SWAPDROPSWAP	77, 78	SYM^	59
SWAPDROPTTRUE	87	SYM^0	59
SWAPDUP	77	SYM<	62
SWAPINCOMP	51	SYM<=	62
SWAPINDEX@	101	SYM<>	62
SWAPLOOP	101	SYMABS	60
SWAPONE	13	SYMACOS	61
SWAPOVER	76, 77	SYMACOSH	61
SWAPOVER#-	15	SYMALOG	61
swappargunrot	69	SYMAND	61
SWAPROT	77, 78	SYMARG	61
SWAPROWS	41	SYMASIN	61
SWAPTRUE	87	SYMASINH	61
SWAPUnDROP	53	SYMATAN	61
SWAPUnNDROP	53	SYMATANH	61
SWITCH2FLOATS	26	symp	5
SWITCHFLOATS	26	Symb>HBuff	154
SWP1+	15, 53	SYMBCMP	10
sym	5	sympn	288
SYM%CH	60	SYMBN	50, 56
SYM%OF	60	SYMBN:	50
SYM%T	60	SYMBNUMSOLVE	288
SYM%>	62	SYMBREAL	10
SYM%>=	62	SYMBSYM	10
SYM%<	62	SYMBUNIT	10
SYM%<=	62	SYMBWHERE	67
SYM%AND	61	SYMCEIL	60

SYMCPCMP	12	SYMREAL	10
SYMCMPREAL	11	SYMREALCMP	11
SYMCMPSYM	12	SYMREALREAL	11
SYMCOLCT	63	SYMREALSYM	11
SYMCOMB	60	SYMRND	60
symcomp	57	SYMSHOW	68
SYMCONJ	60	SYMSIGN	60
SYMCOS	61	SYMSIN	61
SYMCOSH	61	SYMSINH	61
SYMCRUNCH1	57	SYMSQ	61
SYMCRUNCH2	57	SYMSQRT	61
SYMD>R	61	SYMSUM	68
SYMDER	68	SYMSYM	10
SYMDERSTEP	68	SYMSYMB	10
SYMEXP	61	SYMSYMCMP	12
SYMEXPAN	64	SYMSYMREAL	12
SYMEXPM	61	SYMTAN	61
SYMEXPONENT	60	SYMTANH	61
SYMEXT	10	SYMTAYLR	68
SYMFACT	61	SYMTRCN	60
SYMFLOAT==	61	SYMUBASE	61
SYMFLOAT<>	62	SYMUVAL	61
SYMFLOOR	60	SYMWHERE	67
SYMFP	60	SYMXOR	61
SYMID	10	SYMXROOT	60
SYMIDCMP	12	SYNTAXERR	85
SYMIDEXT	12	Sys@	102
SYMIDLIST	12	SYSCONTEXT	105
SYMIDREAL	12	SysDisplay	143
SYMIFTE	68	SysErrorTrap	288
SYMIM	60	SysITE	95
SYMINTEG	65	SysMenuCheck	130
SYMINV	60	SysNib1	272
SYMIP	60	SysNib10	273
SYMISOL	68	SysNib11	273
SYMLAM	10	SysNib12	273
SYMLIST	10	SysNib13	273
SYMLN	61	SysNib14	274
SYMLNP1	61	SysNib15	274
SYMLOG	61	SysNib16	274
SYMMANT	60	SysNib17	274
SYMMAX	59	SysNib18	274
SYMMIN	59	SysNib19	274
SYMMOD	59	SysNib2	272
SYMNEG	60	SysNib20	275
SYMNOT	60	SysNib3	272
SYMOB	10	SysNib4	272
SYMOR	61	SysNib5	272
SYMPERM	60	SysNib6	272
SYMQUAD	68	SysNib7	273
SYMR>D	61	SysNib8	273
SYMRE	60	SysNib9	273

tok7	31	TWENTYEIGHT	6
tok8	31	TWENTYFIVE	6
tok8cktrior	36	TWENTYFOUR	6
tok8trior	36	TWENTYNINE	6
tok9	31	TWENTYONE	6
tokanglesign	31	TWENTYSEVEN	6
tokCTGROB	31	TWENTYSIX	6
tokCTSTR	31	TWENTYTHREE	6
tokDER	31	TWENTYTWO	6
tokESC	31	TWO	5
tokexponent	31	TWO{ }N	51
toklparen	30	TWODROPNULL\$	32
tokquote	31	TYPE	122
tokrparen	30	TYPEARRAY?	122
toksharp	31	TYPEARRY?	122
tokSIGMA	31	TYPEARRY@	41
tokSQRT	31	TYPEBINT?	123
tokUNKNOWN	32	TYPECARRY?	122
tokuscore	31	TYPECHAR?	123
tokWHERE	31	TYPECMP	12
toLEN_DO	101	TYPECMP?	122
top&	53	TYPECOL	12
top&Cr	50	TYPECOL?	123
top&pshtop&	53	TYPECSTR?	122
TOP16	149	TYPEEREL	12
TOP8	148	TYPEEXT	12
TopERow?	289	TYPEEXT?	122
TOPLINE	281	TYPEGROB?	122
TOPROW	149	TYPEHSTR?	122
TOSRRP	73	TYPEIDNT	12
TOTEMPOB	106	TYPEIDNT?	122
TOTEMPOBADJ	106	TYPELAM	12
TOTEMPSWAP	106	TYPELAM?	122
TOUCHTAB	281	TYPELIST	12
TrackAct	281	TYPELIST?	122
TrackMASK	272	TYPERARRY?	122
TRCNYM	60	TYPEREAL	12
TRCXY	21	TYPEREAL?	122
TRPACKETFAIL	116	TYPEROMP?	122
TRUE	87	TYPERRP	12
TRUE'	98	TYPERRP?	123
TrueFalse	87	TYPESYMB	12
TRUEFALSE	87	TYPESYMB?	122
TrueTrue	87	TYPETAGGED?	122
TST15	289		
TTHIRTYSIX	6		
TURNMENUOFF	142		
TURNMENUON	142		
TurnOff	112		
TurnOffKey	289		
TWELVE	5		
TWENTY	6		

U

U>nbr	46
U>NCQ	46
UART?	117
uart_buf_end	276
uart_buf_st	276
uart_buffer	276
uart_error	276
uart_handshk	276
uart_modes	276
uart_parity	276
uart_timeout	276
UARTBUFLEN	117
UARTxcp	117
UCursor	289
UM#?	47
UM%	46
UM%CH	46
UM%T	46
um*	45
UM*	46
UM-	46
um/	45
UM/	46
UM=?	47
UM+	46
UM>=?	47
UM>?	47
UM>U	46
um^	45
UM<=?	47
UM<?	47
UMABS	47
UMCEIL	47
UMCHS	47
UMCONV	46
UMCOS	47
umEND	45
UMFACT	46
UMFLOOR	47
UMFP	47
UMINV	47
UMIP	47
UMMAX	47
UMMIN	47
UMOPER:	47
umP	45
UMRND	47
UMSI	46
UMSIGN	47
UMSIN	47
UMSQ	47
UMSQRT	47
UMTAN	47
UMTRC	47
UMU>	46
UMXROOT	47
Unbr>U	46
UNCOERCE	19
UNCOERCE%%	19
UNCOERCE{}2	19
UNCOERCE2	19
uncrunch	57
undo	81
UNDO_OFF	112
UNDO_ON	112
UNDO_ON?	112
UNDOMASK	272
UNIT>\$	46
unitob	5
UnLockAlpha	148
UNROLL	78
unrot	52
UNROT	78
unrot1	54
UNROT2DROP	77, 78
UNROT2DROP%0	19
UNROTDROP	77, 78
UNROTDUP	78
UNROTOVER	78
UNROTSWAP	77, 78
UNROTSWAPDROP	78
UNTIL	100
UobROT	54
UPDIR	104
USER\$>TAG	40
UserFlags	271
UserFlagStat	289
UserITE	95
UserKeys	281
UserKeys!	126
UserKeys?	126
UserKeysStat	289
USEROB	270
UStackDepth	77
UTTYPEEXT0? (~UTTYPEEXT0?)	289
UTVUNS1Arg (~UTVUNS1Arg)	289

V

VARSIZE	113
VDISP	270
VDISP1	270
VDISP2	270
VDISP3	270
VerifyTOD	107
VERSTRING	117
VERSTRING?	117
VERYSLOW	107
VERYVERYSLOW	107
VIEWLEVEL	281
ViewLevel1	139
VLM	289
VSCALE	160

W

w->W	262
wait	107
Wait/GetKey	125
WaitForKey	125
WaitTbz0	289
Warmstart	289
WHERE1	68
WEREN	68
WHILE	100
WINDOW#	150
WINDOWBOT?	150
WINDOWCORNER	149
WINDOWDOWN	149
WINDOWLEFT	149
WINDOWLEFT?	150
WINDOWRIGHT	149
WINDOWRIGHT?	150
WINDOWTOP?	150
WINDOWUP	149
WindowXY	289
WINDOWXY	149
WIPEOUT	261
WithHidden	105
WORDSIZE	38
WSLOG	113
WSLOGN	113

X

x#?	249
x%	250
x%CH	174
x%T	234
x'	200

x*	251
x*H	226
x*W	226
x-	252
x->Q	217
x->QPI	217
x->TAG	235
x/	253
x=	254
x==	254
X@	289
x+	251
x>	255
x>=?	249
x>>	200
x>>ABND	200
x>ARRAY	165
x>GROB	195
x>HMS	197
x>LCD	201
x>LIST	203
x>NUM	209
x>STR	233
x>UNIT	241
x>V2	243
x>V3	243
x^	247
x<	253
x<=?	248
x<<	200
xABS	162
xACK	162
xACKALL	162
xACOS	163
xACOSH	163
xALG->	250
xALOG	163
xAND	163
xAPPLY	164
xARC	164
xARCHIVE	164
xARG	165
xARRAY>	165
xASIN	165
xASINH	166
xASN	166
xASR	167
xATAN	167
xATANH	168
xATTACH	168
xAUTO	168
xAXES	169

xB>R	173	xDELKEYS	181
xBAR	169	xDEPND	181
xBARPLOT	170	xDEPTH	182
xBAUD	171	xDER	282
xBEEP	171	xDET	182
xBESTFIT	171	xDETACH	182
xBIN	171	xDIR	182
xBINS	172	xDISP	182
xBLANK	172	xDO	182
xBOX	172	xDOERR	183
xBUFLen	172	xDOT	183
xBYTES	173	xDRAW	183
xC>PX	179	xDRAX	183
xC>R	179	xDROP	184
xCASE	173	xDROP2	184
xCEIL	173	xDROPN	184
xCENTR	173	xDTAG	184
xCF	174	xDUP	184
xCHR	174	xDUP2	184
xCKSM	174	xDUPN	184
xCLEAR	175	xELSE	185
xCLKADJ	175	xENDDO	185
xCLLCD	175	xENDTIC	200
xCLOSEIO	175	xENG	186
xCLSIGMA	175	xEQ>	186
xCLUSR	175	XEQ>ARRAY	44
xCNRM	176	XEQ>ARRY	44
xCOLCT	176	XEQ>VECTOR	44
xCOMB	176	XEQAsnKey	126
xCON	176	XEQDelKeys	126
xCONIC	177	XEQINTEG	65
xCONJ	177	XEQINTEGID	65
xCONSTANTe	185	XEQIOBACKUP	117
xCONT	177	XEQLIST>	51
xCONVERT	177	XEQORDER	104
xCORR	178	XEQPGDIR	104
xCOS	178	XEQPOS\$	33
xCOSH	178	XEQPURGEPICT	159
xCOV	178	XEQRCL	102
xCR	179	XEQRclKeys	126
xCRDIR	179	XEQRCWS	111
xCROSS	179	XEQSETLIB	73
xD>R	185	XEQSHOWLS	68
xDATE	180	XEQSTOID	103
xDATE+	180	XEQStoKey	103
xDDAYS	180	XEQSTWS	111
xDEC	180	XEQSUB\$	34
xDECR	180	XEQSYMDERCON	68
xDEFINE	181	XEQSYMDERSTEP	68
xDEG	181	XEQSYMWHERE	67
xDELALARM	181	XEQTYPE	122
xDELAY	181	xERASE	186

xERRO	187	xINDEP	199
xERRM	187	xINPUT	199
xERRN	187	xINTEGRAL	282
xERRTHEN	236	xINV	199
xEVAL	187	xIP	199
xEXP	188	xISOL	200
xEXPAN	188	xKERRM	200
xEXPFIT	188	xKEY	200
xEXPM	189	xKGET	200
xFACT	250	xKILL	201
xFC?	189	xLABEL	201
xFC?C	189	xLAST	201
xFCNAPPLY	282	xLCD>	201
XFERFAIL	12	xLIBS	202
xferfailerr	257	xLINE	202
xFINDALARM	189	xLINFIT	202
xFINISH	190	xLIST>	202
xFIX	190	xLN	203
xFLOOR	190	xLNP1	203
xFORMUNIT	200	xLOG	203
xFP	191	xLOGFIT	203
xFREE	191	xLR	204
xFREEZE	191	xMANT	204
xFS?	192	xMATCHDN	204
xFS?C	191	xMATCHUP	204
xFUNCTION	192	xMAX	205
xGET	192	xMAXR	205
xGETI	193	xMAXSIGMA	205
xGOR	194	xMEAN	205
xGRAD	195	xMEM	206
xGRAPH	212	xMENU	206
xGROB	195	xMERGE	206
xGXOR	195	xMIN	206
xHALT	195	xMINR	207
xHEX	196	xMINSIGMA	207
XHI	9	XmitSrcvTOut	281
XHI-1	9	xMOD	207
xHISTOGRAM	196	xNEG	208
xHISTPLOT	196	xNEGNEG	96
xHMS-	196	xNEWOB	208
xHMS+	196	xNEXT	208
xHMS>	197	xNOT	208
xHOME	197	xnsgeneral	54
xi	197, 198	xNSIGMA	208
xIDN	197	xNUM	209
xIF	197	xOBJ>	209
xIFEND	185	xOCT	209
xIFERR	198	xOFF	210
xIFT	198	xOLDPRT	210
xIFTE	198	xOPENIO	210
xIM	198	xOR	210
xINCR	199	XOR	88

XOR\$	35	xRDZ	221
xORDER	210	xRE	221
xOVER	211	xRECN	221
xPARAMETRIC	211	xRECV	221
xPARITY	211	xREPEAT	222
xPATH	211	xREPL	222
xPDIM	211	xRES	222
xPERM	211	xRESTORE	222
xPGDIR	212	xRL	223
xPI	248	xRLB	223
xPICK	212	xRND	223
xPICT	212	xRNRM	223
xPIX?	212	xROLL	223
xPIXOFF	213	xROLLD	224
xPIXON	213	xROOT	224
xPKT	213	xROT	224
xPMAX	213	xRPN->	250
xPMIN	213	xRR	224
xPOS	213	xRRB	224
xPR1	214	xRSD	224
xPREDIV	218	xRULES	225
xPREDV	214	xSAME	225
xPREDX	214	xSBRK	225
xPREDY	214	xSCALE	225
xPRLCD	215	xSCATRLOT	226
xPROMPT	215	xSCATTER	226
xPRST	215	xSCI	226
xPRSTC	215	xSCLSIGMA	227
xPRVAR	215	xSCONJ	227
xPURGE	216	xSDEV	227
xPUT	216	xSEND	227
xPUTI	216	xSERVER	227
xPVAR	216	xSETDATE	180
xPVIEW	217	xSETTIME	236
xPWRFIT	217	xF	228
xPX>C	217	xSHOW	228
xQUAD	218	xSIGMA-	248
xQUOTE	218	xSIGMA+	248
xR>B	225	xSIGMACOL	176
xR>C	225	xSIGMALINE	202
xR>D	225	xSIGN	228
xRAD	218	xSILENT'	98
xRAND	218	xSIN	228
xRCEQ	219	xSINH	228
xRCL	219	xSINV	229
xRCLALARM	219	xSIZE	229
xRCLF	219	xSL	229
xRCLKEYS	220	xSLB	229
xRCLMENU	220	xSNEG	230
xRCLSIGMA	220	xsngeneral	54
xRCWS	220	xSQ	230
xRDM	220	xSQRT	247

Z

Z-BOX	289	ZEROOVER	13
ZERO	5	ZEROSWAP	13
ZERO_DO	100	ZEROZERO	12
ZEROFALSE	13	ZEROZEROONE	13
ZEROISTOPSTO	101	ZEROZEROTWO	13
		ZEROZEROZERO	13