# Programming in System RPL

### Eduardo M Kalinowski Carsten Dominik

#### Preface to the Second Edition

Back in 1998, when the first edition of this book was released, it proved to be a good tutorial for new learners of System RPL, and also a good rerefence for more experienced programs.

However, there was still room for improvement. And when the HP49G calculator was released, the need for a second edition of this book was even greater, because no document describing all its new features existed.

For these reasons, we have put together this new edition. Those who have used the first edition of this document will find many changes and improvements, and also some 400 additional pages:-). The structure of the book has been totally changed, with the tutorial and reference parts merged. All the text has been revised and corrected. Some chapters were completely rewritten in order to make them easier to understand and more useful. There are also new chapters, describing new HP49 features, and also about things that were not described in the first edition.

We hope this book is a valuable resource for those that already knew System RPL on the HP48 and wanted more information on the new HP49 features, and for those that want to start learning System RPL in order to discover more of the power of the HP49.

April 24, 2002

Eduardo de Mattos Kalinowski Carsten Dominik

#### Preface to the First Edition

The programming features of the HP48 graphical calculator are very powerful. They allow you to do virtually anything. However, the documented programming functions, that are directly accessible to the user (the user language), is not everything the calculator can do.

There is another language: the System language. The User language is a subset of the System one, with just some commands and just a fraction of its power. However, the System language is not well documented. The existing documents on that subject are turned to someone who already knows it; they are just listings of the commands with some brief descriptions. Once you already know the language, even the brief descriptions can be left out, and those documents are really a very good source of information. But how does one *learn* System RPL?

The purpose of this book is exactly that: to be a way for someone who has already learned User RPL (if you have not yet, learn it before, then come back to this), and wants to learn the *real* power of the calculator.

July 12, 1998

Eduardo de Mattos Kalinowski

### **Acknowledgments**

This work could not have been accomplished without the help of many people. Firstly, we would like to thank Mika Heiskanen for his entry point list by subject. That document was an important source of information. His JAZZ library with its disassembler and other functions was also an indispensable gateway to the HP48 ROM code.

We would also like to thank the ACO team, for developing the HP49, and for providing us with additional information for writing some of the chapters in this book.

Our thanks go also to Wolfgang Rautenberg, who reviewed the first edition throughly and gave many suggestions for this present edition.

Many other people helped during the two editions of the book:

Al Arduengo	Joe Horn	Alberto Zamora Oyace
Jean-Yves Avenard	Werner Huysegoms	<b>Bernard Parisse</b>
Jurjen N.E. Bos	David Kastrup	Richard Pascal
Carlos Bourlot	Dan Kirkland	James M Prange
Sune Bredahl	Piotr Kowalewski	Thomas Rast
Jonathan Busby	Daniel Lidström	Eric Rechlin
Cyrille de Brébisson	<b>Andreas Matthias</b>	Melissa Reid
Stefan Ehlen	Andreas Möller	Ricardo Blasco Serrano
Len Fellman	Denis Martinez	Gerald Squelart
Peter Geelhoed	Hakim Mazouz	Pierre Tardy
Christoph Gießelink	Christian Meland	Jernej Zajc
Raymond Hellstern	John H Meyers	Adam Zwierko
Jordi Hidalgo	Heiko Oberdiek	

If we forgot someone, please forgive us, and be sure we are grateful anyway.

#### **Disclaimer**

This document is © by Eduardo M Kalinowski and Carsten Dominik.

It is distributed in the hope it will be useful for those who want to learn System RPL or want a reference about it, but it is provided "as is", without warranty of any kind, either expressed or implied. In no event, we shall be liable to you for damages, including any general, special, incidental or consequential damages caused directly or indirectly by the use of the information provided here. We do not assure that any information here is right. Use it at your own risk.

This document may be distributed only if in its whole, unmodified form; and if you do not charge anything for it (except nominal copying fees). Otherwise, we will want our share!

The latest version of this document and its errata can be found at the homepage http://move.to/hpkb.

### **Short Contents**

Pr	reface to the Second Edition	i		
Pr	Preface to the First Edition			
Ac	eknowledgments	iii		
Di	sclaimer	iv		
1	Introduction	1		
Ι	HP49 Objects	7		
2	Binary Integers (BINTS)	9		
3	Real Numbers	27		
4	Complex Numbers	36		
5	Integers (ZINTS)	39		
6	Characters and Strings	40		
7	Hex Strings	56		
8	Identifiers	60		
9	Tagged Objects	61		
10	Arrays	63		
11	Composite Objects	67		
12	Meta Objects	75		
13	Unit Objects	80		

SHORT CONTENTS
----------------

•	
771	
V I	

14 Symbolics	85	
5 Graphics Objects (Grobs)		
16 Library and Backup Objects	99	
II General System RPL Entries	103	
17 Stack Operations	105	
18 Temporary Environments	111	
19 Runstream Control	120	
20 Conditionals	132	
21 Loops	147	
22 Error Handling	153	
23 The Virtual Stack	159	
24 Memory Operations	165	
25 Time and Alarms	172	
26 System Functions	175	
27 Serial Communications	180	
28 The HP49 Filer	182	
III Input and Output	189	
29 Checking for Arguments	191	
30 Keyboard Control	202	
31 Using InputLine	209	
32 The Parameterized Outer Loop	213	

SHORT CONTENTS	vii
33 Using the HP49 Browser	225
34 Using the HP48 Browser	235
35 Creating Input Forms	248
36 The Display	268
37 The Menu	284
38 Programming the HP49 Editor	295
39 Plotting	315
IV The HP49 CAS	319
40 Introduction to the HP49 CAS	321
41 Type Checking and Conversion	325
42 Integers	327
43 Matrices	336
44 Expression Manipulation	347
45 Symbolic Meta Handling	359
46 Polynomials	371
47 Root Finding	381
48 Calculus Operations	386
49 Summation	393
50 Modular Operations	395
51 Sign Tables	400
52 Errors	403
53 CAS Configuration	405

viii	SHORT CONTENTS
54 CAS Menus	409
55 Internal Versions of User RPL Commands	411
56 Miscellaneous	417
V Appendices	427
A Development Tools	429
B Creating Libraries	447
C User RPL Commands	453
D Library 256 and EXTABLE	478
E Error Messages	480
VI Index	497
F Entries sorted by Name	499
G Entries sorted by Address	557

### **Contents**

Pı	refac	reface to the Second Edition			
Pı	reface to the First Edition				
A	ckno	wledgr	nents	iii	
Di	iscla	imer		iv	
1	Int:	roducti	i <b>on</b> 'irst System RPL Program	1 3	
	1.1		the Entries Listing	5	
Ι	HP	949 Obj	jects	7	
2	Bin	ary Int	tegers (BINTS)	9	
	2.1	Refere	nce	10	
		2.1.1	Built-in Binary Integers	10	
		2.1.2	Pushing Several BINTs	21	
		2.1.3	Conversion	22	
		2.1.4	Arithmetic Functions	23	
		2.1.5	Tests	25	
3	Rea	al Num	bers	27	
	3.1	Refere	nce	28	
		3.1.1	Built-in Real Numbers	28	
		3.1.2	Built-in Extended Real Numbers	30	
		3.1.3	Stack Manipulation Combined with Reals	31	
		3.1.4	Conversion	31	
		3.1.5	Real Functions	31	
		3.1.6	Extended Real Functions	33	
		3.1.7	Tests	35	
4	Cor	nnley N	Numbers	36	

X CONTENTS

	4.1	Referen 4.1.1 4.1.2 4.1.3 4.1.4	ace       3         Builtin Complex Numbers       3         Conversion       3         Functions       3         Tests       3	6 7 7
5	Inte	egers (Z	INTS) 3	9
6	Cha	racters	and Strings 4	0
	6.1	Referen	ice	1
		6.1.1	Built-in Characters	1
		6.1.2	Built-in Strings	3
		6.1.3	Built-in Strings with Stack Manipulation 4	6
		6.1.4	Conversion	6
		6.1.5	Management	7
		6.1.6	Parsing Strings	0
		6.1.7	Decompilation	1
		6.1.8	String Tests	5
7	Hex	Strings	s 50	6
	7.1	Referen		6
		7.1.1	Conversion	6
		7.1.2	General Functions	7
		7.1.3	Tests	8
8	Idei	ntifiers	6	0
9	Tag	ged Obj	ects 6	1
U	_	Referen		
10	Arra		6	9
10		ays Referen		_
	10.1	10.1.1	General Functions	
		10.1.1	Conversion	
		10.1.2	-	
		10.1.5	Statistics	O
11		_	Objects 6	
	11.1	Referen		
		11.1.1	General Operations	
		11.1.2	Building	
		11.1.3	Exploding	1

CONTENTS	X
0011121110	21

	11.1.4	Lists	72
	11.1.5	Secondaries	73
<b>12</b>	Meta Objec	ets	<b>75</b>
	12.1 Referen	nce	75
	12.1.1	Stack Functions	75
	12.1.2	Combining Functions	76
	12.1.3	Meta and Object Operations	77
	12.1.4	Other Operations	77
13	Unit Objec	ts.	80
	13.1 Referen		81
	13.1.1	Creating Units	81
	13.1.2	General Functions	82
	13.1.3	Arithmetic Functions	83
	13.1.4	Tests	83
14	Symbolics		85
	14.1 Referen	nce	86
	14.1.1	General Operations	86
	14.1.2	Other Functions	88
	14.1.3	Meta Symbolics Functions	88
15	Graphics C	Objects (Grobs)	89
	15.1 Referen		90
	15.1.1	Built-in Grobs	90
	15.1.2	Dimensions	90
	15.1.3	Grob Handling	90
	15.1.4	Greyscale Graphics	93
	15.1.5	Creating Menu Label Grobs	95
	15.1.6	Converting Strings to Grobs	96
	15.1.7	Creating Grobs from Other Objects	98
16	Library an	nd Backup Objects	99
	16.1 Referen		99
	16.1.1	Port Operations	99
	16.1.2	<del>-</del>	100
	16.1.3		100
	16.1.4		101

xii CONTENTS

II	General System RPL Entries	103
17	Stack Operations	105
	17.1 Reference	. 105
18	<b>Temporary Environments</b>	111
	18.1 Named Local Variables	. 111
	18.2 Unnamed Local Variables	. 112
	18.3 Nested Temporary Environments	. 113
	18.4 Other Ways of Binding	. 114
	18.5 Reference	. 116
	18.5.1 Builtin IDs and LAMs	. 116
	18.5.2 Conversion	. 116
	18.5.3 Temporary Environments Words	. 116
19	Runstream Control	120
	19.1 Some Concepts	. 120
	19.2 Runstream Commands	. 122
	19.3 Some Examples	. 124
	19.4 Reference	. 128
	19.4.1 Quoting Objects	. 130
	19.4.2 Skipping Objects	. 131
20	Conditionals	132
	20.1 Tests	. 132
	20.2 IfThenElse	. 133
	20.3 Case	. 133
	20.4 Reference	. 135
	20.4.1 Boolean Flags	. 135
	20.4.2 General Tests	. 137
	20.4.3 True/False Tests	
	20.4.4 Binary Integer Tests	. 140
	20.4.5 Real and Complex Number Tests	. 142
	20.4.6 Meta Object Tests	
	20.4.7 General Object Tests	
	20.4.8 Miscellaneous	. 146
21	Loops	147
	21.1 Indefinite Loops	
	21.1.1 How Indefinite Loops Work	. 148

xiii

	21.2 Definite Loops	149
	21.2.1 How a DO Loop Works	150
	21.3 Reference	
	21.3.1 Indefinite Loops	
	21.3.2 Definite Loops	
22	T. T. W.	
22	8	153
	11 0	153
		154
	8	155
	22.3 Reference	
	22.3.1 General Words	156
	22.3.2 Error Generating Words	157
23	The Virtual Stack	159
		160
	2012 10010201100 1 1 1 1 1 1 1 1 1 1 1 1	100
24	Memory Operations	165
	· ·	166
		166
	24.1.2 Directories	
	24.1.3 The Hidden Directory	
	24.1.4 Temporary Memory	
	= 1.2.1	
<b>25</b>	Time and Alarms	<b>172</b>
	25.1 Reference	172
	25.1.1 Alarms	174
00		1 ==
26		175
	26.1 Reference	
	26.1.1 User and System Flags	
	26.1.2 General Functions	178
27	Serial Communications	180
	27.1 Reference	180
90	The HP49 Filer	100
48		182
	8	182
	- 11 8	182
	_ 8	182
	28.1.3 The Filer_List Argument	183

xiv CONTENTS

		28.1.3.1 28.1.3.2	Name_Item	 183 183
		28.1.3.3	Action_Item	
		28.1.3.4	ExtraProgram_Item	186
		28.1.3.5	Key_Shortcut	187
	28.2 Refere	ence		 188
II	I Input a	nd Outp	out	189
29	Checking	for Argu	ments	191
	29.1 Numb	er of Argu	aments	 191
	29.2 Argun	nent Type		 193
	29.2.1	Exampl	les	 195
				196
	29.3.1	Type Cl	hecking	 198
30	Keyboard	Control		202
	30.1 Key Le	ocations .		 202
	30.2 Waitin	ng for a Ke	ey	 204
	30.3 Refere	ence		 205
	30.3.1	Convert	ting Keycodes	 205
	30.3.2	Waiting	g for Keys	 205
	30.3.3	The AT	TN Flag	 207
	30.3.4	Bad Ke	ys	 207
	30.3.5	User Ke	eys	 208
31	Using Inp	utLine		209
	31.1 Menu	Key Assig	gnments	 211
	31.2 An Ex	ample	- 	 211
	31.3 Refere	ence		 212
<b>32</b>	The Paran	neterized	d Outer Loop	213
			Outer Loop Words	 214
			nments	216
			gnments	
			pended Environments	
			tion	218

CONTENTS xv

	32.8 An Example	218
	32.9 Reference	223
33	Using the HP49 Browser 2	225
•	33.1 The Choose Items meta	
	33.2 The Title String	
	33.3 The Initially Selected Item	
	33.4 The Message Handler	
	33.5 The Browser and Lams	
	33.6 Accessing the Selected Item	
	33.7 Saving and Restoring the Screen	
	33.8 An Example	
	33.9 Reference	
<b>34</b>	- 1 <b>8</b> · 1 · 1 · 1 · 1 · 1 · 1 · 1 · 1 · 1 ·	235
	34.1 The ::Appl Parameter	
	34.2 The \$Title Parameter	
	34.3 The ::Converter Parameter	
	34.4 The {}Items Parameter	
	34.5 The Init Parameter	
	34.6 Typical Browser Usage	
	34.7 An Example	
	34.8 Reference	
	34.8.1 NULLLAMs Used by the Browser	247
35	Creating Input Forms 2	248
	35.1 Label Definitions	249
	35.2 Field Definitions	249
	35.3 Label and Field Counts	251
	35.4 Message Handlers	
	35.5 The Title	
	35.6 Results Of The Input Form	
		253
	<u>.</u>	257
		257
		262
		262
	- ·	262
	35.8.2.3 IfMsgNewField — 2	
	35.8.2.4 IfMsgGetFocus — 3	

xvi CONTENTS

	;	$35.8.2.5$ IfMsgGetFieldValue — $4 \dots \dots$	 	263
	;	35.8.2.6 If MsgSetFieldValue — $5 \ldots \ldots$	 	263
	;	35.8.2.7 If MsgGetFieldGrob — 6 $\dots \dots$	 	263
	;	$35.8.2.8  If Msg SetFirstField 7 \ldots \ldots \ldots$	 	264
	;	35.8.2.9 IfMsgFieldReset — $10 \dots \dots$	 	264
	;	35.8.2.10 IfMsgGetMenu — 11 $\dots \dots$	 	264
	;	35.8.2.11 IfMsgGet3KeysMenu — 12	 	265
	;	35.8.2.12 IfMsgCancel — 13	 	265
		35.8.2.13 IfMsgCancelKey — 14 $\dots$		
	;	35.8.2.14 IfMsgOK — 15	 	265
	;	$35.8.2.15~IfMsgKeyOK 16\ldots\ldots\ldots$	 	266
	;	35.8.2.16 IfMsgChoose — 17	 	266
	;	35.8.2.17 IfMsgType — 18 $\dots$	 	266
		35.8.2.18 IfMsgCalc — 19		
	;	$35.8.2.19~IfMsgNewCommandLine 20~\dots~.$	 	267
	;	$35.8.2.20~If MsgOld Command Line 21~\dots~.$	 	267
	;	$35.8.2.21~IfMsgCommandLineValid — 22~\dots~.$	 	267
	;	$35.8.2.22~IfMsgDecompEdit 23\ldots\ldots\ldots$	 	267
	;	35.8.2.23 IfMsgNextChoose — 24	 	267
	;	35.8.2.24 IfMsgEdit — 25	 	267
0.0 TN	D: 1			000
	e Displa	· <del>-</del>		268
		y Organization		
		ring the Display		
		olling Display Refresh		
		ng the Display		
36.		ying Text		
	36.5.1	v		
		Minifont		
200		Displaying Warnings		
36.		nce		
	36.6.1	F . 7 . 8		
		Preparing the Display		
		Immediate Refresh		
	36.6.4	Controlling Display Refresh		275
	36.6.5	Clearing the Display		
	36.6.6	Annunciator and Modes Control		277
	36.6.7	Window Coordinates		279
	36.6.8	Scrolling the Display		279
	36.6.9	Displaying Objects	 	280

CONTENTS	xvi
CONTENTS	AVI

	3	36.6.10	Displaying Text	281
	3	36.6.11	Fonts	283
<b>37</b>	The N	Menu		284
	37.1 N	Menu F	ormat	285
	37.2 I	Menu P	roperties	286
	37.3 I	Referen	ce	288
	9	37.3.1	Menu Properties	288
	9	37.3.2	Building Menus	291
	9	37.3.3	Menu Display	292
	9	37.3.4	Displaying Menu Labels	293
	3	37.3.5	General Entries	293
38	Prog	rammi	ng the HP49 Editor	295
	38.1 7	<b>Fermin</b> o	ology	295
	38.2 I	Exampl	es	296
	38.3 I	Executi	ng External Commands in the Editor	299
	38.4 I	Referen	ce	300
	3	38.4.1	Status	300
	3	38.4.2	Inserting Text	301
	3	38.4.3	Deleting Text	302
	9	38.4.4	Moving the Cursor	304
	9	38.4.5	Selection, Cut and Paste, the Clipboard	306
	9	38.4.6	Search and Replace	308
	9	38.4.7	Evaluation	309
	9	38.4.8	Starting the Editor	310
	3	38.4.9	Miscellaneous	311
9	Plott	ing		315
	39.1 F	Referen	ce	316
IV	The	e HP4	9 CAS	319
10	Intro	ductio	on to the HP49 CAS	321
			ns with These Chapters	
			ic Objects	
			Examples	
<b>41</b>	Type	Check	ing and Conversion	325
			ce	

xviii CONTENTS

<b>42</b>	Integers		<b>327</b>
	42.1 Referen	nce	327
	42.1.1	Built-in Integers	327
	42.1.2	Conversion Functions	327
	42.1.3	General Integer Operations	329
	42.1.4	Integer Factorization and Prime Numbers	
	42.1.5	Gaussian Integers	
	42.1.6	Integer Tests	
43	Matrices		336
	43.1 Referen	nce	337
	43.1.1	Creating and Redimensioning Matrices	
	43.1.2	Conversion	
	43.1.3	Tests	
	43.1.4	Calculations with Matrices	
	43.1.5	Linear Algebra and Gaussian Reduction	
	43.1.6	Linear System Solver	
	43.1.7	Other Matrix Operations	
	43.1.8	-	
44	Expression	Manipulation	347
	44.1 Referen	<del>-</del>	347
	44.1.1	Basic Operations and Function Application	347
	44.1.2	-	
	44.1.3		
	44.1.4	Collection and Expansion	
	44.1.5	Trigonometric Transformations	
	44.1.6	Division, GCD and LCM	357
45	Symbolic M	Meta Handling	359
	45.1 Referen	_	359
	45.1.1	Basic Expression Manipulation	
	45.1.2	Basic Operations and Function Application	
	45.1.3	Trigonometric and Exponential Operators	365
	45.1.4	Infinity and Undefs	367
	45.1.5	Expansion and Simplification	368
	45.1.6	Tests	370
46	Polynomia	ls	371
	46.1 Referen	nce	371

CONTENTS	xix

	46.1.1	Computation with Polynomials	371
	46.1.2	Factorization	373
	46.1.3	General Polynomial Operations	377
	46.1.4	Tests	380
47	Root Findi	ng	381
	47.1 Referen		
	47.1.1		
48	Calculus O	nerations	386
-0		nce	
		Limits and Series Expansion	
	48.1.2	Derivatives	
	48.1.3	Integration	
		Partial Fractions	
	48.1.5	Differential Equations	
	48.1.6	Laplace Transformation	
	40.1.0	Laplace Transformation	0 <i>9</i> 2
<b>49</b>	Summation	1	393
	49.1 Referen	nce	393
50	Modular O	perations	395
		nce	
		Modulo Operations	
51	Sign Tables	S	400
	_	nce	400
<b>52</b>	Errors		403
	52.1 Referen	nce	403
<b>53</b>	CAS Config	guration	405
		nce	405
54	CAS Menus		409
<i>J</i>		nce	
	5 1,1 10010161		100
<b>55</b>	Internal Ve	ersions of User RPL Commands	411
	55.1 Referen	nce	411
56	Miscellane	ous	417

XX CONTENTS

	56.1	Reference	417
		56.1.1 Verbose Mode Display Routines	417
		56.1.2 Evaluation	417
		56.1.3 Conversion	418
		56.1.4 Qpi	418
		56.1.5 Infinity	419
		56.1.6 Built-In Constants	420
		56.1.7 List Application	
		56.1.8 Irrquads	
		56.1.9 Miscellaneous	422
V	Ap	pendices 4	127
A	Dev	relopment Tools	<b>129</b>
	A.1	The Entry Points Library	430
		About Key Assignments	
	A.3	Hacking Tools	
		A.3.1 Operating Tools for the HP49	
	A.4	The Compiler	
		A.4.1 MASD and the Different Kinds of Entries	
		A.4.2 MASD's Special Features	
		A.4.2.1 Unnamed Local Variable Binding	
		A.4.2.2 Including Source Files	
	A.5	Disassembly	
		A.5.1 Using Nosy	
	A C	A.5.2 Using CQIF?	
		The Editor, and Emacs	
		Debugging	
		JAZZ for the HP49	
В			<b>147</b>
		The Special Variables	
	B.2	The Library Message Handler	
		B.2.1 Menu Extensions	
		B.2.2 Online Help for Libary Commands	
		B.2.3 The Library Menu Message	452
C			<b>453</b>
	U.1	Reference	453

CONTENTS	
D Library 256 and EXTABLE	478
D.1 The Development Library 256	. 478
D.2 The EXTABLE Library	. 479
E Error Messages	480
VI Index	497
F Entries sorted by Name	499
G Entries sorted by Address	557

## Chapter 1 Introduction

If you know how to create programs in User RPL (if you do not, you should learn it before you continue reading this book), then you only know part of what the HP49G calculator can do. The System RPL programming language gives you power to do many things which you could not even imagine. For example, in System RPL you can handle all object types available. User RPL only gives access to some of them. Or you can do math with 15-digit accuracy, use arrays with non-numeric elements, and much more. System RPL can also be used to do the same things as a User RPL program would do, but much faster.

But before we start talking of System RPL, let us go back to User RPL to explain how it really works. We know you are anxious to start with the big thing right now, but the following information is important for a good understanding of System RPL.

HP49 programs (both User and System) are not stored internally using the names of the commands. Only the addresses of the objects are stored. Each of these addresses takes only 2.5 bytes (or more, if the address is a rompointer or flashpointer). When a program is run, the only thing that is actually done is a kind of "gosub" to that address. This way of storing programs serves two purposes. 2.5 bytes is less than the name of most commands, so the program needs less memory. And execution of the program is much faster since during execution, looking up the addresses of names is no longer necessary.

Most of the times, the address points to another program with more jumps to other programs with more jumps, and so on... The calculator keeps track of holding the address to which jump back, and you can have as many jumps as necessary without worrying about it. When the called program ends, you return to where you were before. Of course, the jumps must end somewhere, either in a program written in machine language or in an object that just puts itself in the stack (numbers, strings, etc). This is quite similar to the concept of calling a function or sub-routine in high-level languages.

But if the programs are just addresses, how can they be edited? The an-

2 1. Introduction

swer is that the calculator has a table of the User commands' names and their corresponding addresses. So, when you put a User RPL program in the stack, the HP searches the table to get the name of the commands corresponding to the addresses stored in memory, and then displays the program in a readable form. You can then edit it, and after the edition is done the table is searched again for the addresses of the commands entered, and only these addresses are stored in memory. This is why it takes a long time to edit a long User RPL program, but it is also what makes them fast to run.

This all works as long as all the commands have names. Guess what? There are over four thousand commands without names. This is one of the distinctions between User and System RPL. User RPL, the language described in the manual (the « » language), can only access the named commands. (Actually, it can access the unnamed commands via the commands SYSEVAL, LIBEVAL and FLASHEVAL, as long as you know the address of the command. But this is not efficient (except for an occasional use)). System RPL can access all commands.

Because of that, System RPL programs cannot be edited directly. Special tools are needed for that. In Appendix A you will find information about the available tools for writing System RPL programs. Fortunately, all you need is built-in in the calculator, or is in libraries that can be downloaded to the calculator.

Programming in System RPL is more powerful and much faster, because it does no error checking. In System RPL, the programmer must be sure that no error occurs, otherwise a crash might happen. For example, if a command requires two arguments in the stack and they are not there, or if they are not of the type the function requires, a warmstart or even a memory loss could happen. Naturally, there are commands for checking if there are enough arguments, for their types, and for other possible error conditions. The difference is that you probably just need to check if all arguments are present once, when the program starts. You do not need to repeat the check later. In User RPL, every single command has error checking, so tests are done unnecessarily, slowing the program.

At this point, you might be wondering, "if the commands do not have names, how can you program in System RPL?" As said before, all commands have addresses, so you can call the address directly, using a structure like PTR <address>, and whatever is at that address will be executed. But there is an easier way.

The commands have names. The names simply are not stored in the HP49 in the same way the the names of User commands are. But the HP de-

sign team has given them names, and they are stored in the tools for creating System RPL programs. You write a program using those names, and then the System RPL compiler searches the names in the tables, and converts them to addresses. This is called compiling or assembling. Some tools can also do the opposite: convert the addresses into command names. This is called decompiling or disassembling.

Some of the commands are classified as "supported": they are guaranteed to stay at the same memory location in all ROM versions of the calculator, i.e., their address are not going to change, so programmers can use them safely. (Note that this does not mean they will be in the same address in different calculators, such as the HP48 and HP49.) But there are also commands that are classified as "unsupported". For these, there is not guarantee that they will stay at the same address in different ROM versions.

Unsupported commands are not listed in the tables of compilers, so you cannot enter their names and expect to have their address in the resulting program. You have to either call them directly by their address, or name the command yourself. In the entries listings, the names of unsupported entries will be *enclosed in single parenthesis*, like (CURSOR@@).

Note that all unsupported entries listed in this book are, however, stable. It has been indicated by the HP design team that all HP49G addresses in the ranges 025ECh–0B3C7h and 25565h–40000h will very likely not change, even the unsupported commands in these ranges.

Actually, there are three kinds of entries: the description above dealt mainly with normal 2.5-byte addresses, which point directly to some ROM address. Most entries are of this kind. But there are also rompointer and flash-pointer entries. Rompointers point to commands inside a library. Their names start with  $\tilde{\ }$ . Flashpointers, which only exist in the HP49, point to sub-routines inside the flash memory. Their names start with  $\tilde{\ }$ . Appendix A will describe what is necessary in order to use each kind of entries with HP49 compiler.

#### 1.1 Your First System RPL Program

Let us create a very simple System RPL program, and explain it in detail. The program will calculate the area of a circle, given the radius in the stack. See Appendix A for information on how to compile it. If you downloaded the examples file, you will find it with the name first.

4 1. Introduction

Before we start analyzing it, it is important to note that System RPL is case-sensitive, so pi is different from PI, which is different from pI. Be careful when typing. Also, as you might have guessed, everything between ()'s is considered a comment. Lines that have a \* in the first column are also comments.

The first line contains the start of secondary (i.e., program) marker, :: (called DOCOL). The end marker is ; (SEMI).

Following, there is the command CKINOLASTWD. This command checks if there is one argument in the stack, and if there is not, generates a "Too Few Arguments" error. The next command, CK&DISPATCHO, checks the argument type and allows the programmer to do different things for different argument types. Our program only supports one argument type: real numbers (represented here by BINT1, or the number one as a system binary — see Chapter 2). If any other argument type is entered, a "Bad Argument Type" error will be produced. Argument checking is described in detail in Chapter 29.

After that, there is the code to execute if the argument is a real number. Note that the code is between :: and :. This is because only one object is expected after the argument type. Here, this one object is a secondary (subprogram), one kind of composite object: it is only one object, but with other objects inside it. So if we want to evaluate more than one object, they must be included in a secondary. This is similar to enclosing several statments between braces in C or between begin and end in Pascal.

The rest of the program is very simple. The number two is put in the stack, and the radius (entered by the user) is raised to that power.

Finally ,  $\pi$  is put in the stack, and the squared radius is multiplied by it. The stack now contains the area.

This program is 25 bytes long, as opposed to the 20 of the User RPL program « SQ p \* -> NUM ». However, the User RPL version took 0.0156 seconds to calculate (with radius 1). The System RPL took only 0.0019 seconds. Note that, even if this System RPL program is longer than an equivalent in User RPL, this generally does not happen.

#### 1.2 About the Entries Listing

In the following chapters, the stack diagrams use codes to represent each object type. Here is a list of such codes:

Abbreviation	Meaning
ob	any object
1n	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
м, [[]]	matrix
P	polynom, a list of Qs
Q	ZINT or P
meta, oblobn #n	meta object
grob	graphical object
menu	menu: a program or a list
sign	sign table

UserRPL stack diagrams use some additional abbreviations:

## Abbreviation Meaning x, y real, list, generic UserRPL object complex number hex string (User binary integer)

6 1. Introduction

Abbreviation	Meaning
$\theta$	angle (a real number)
m, n	integer (ZINT or real)
date	date in DD.MMYYYY or MM.DDYYYY format
name	global name
prog, prg	program
f, func	function
F	integral of f

## Part I

# HP49 Objects

## **Chapter 2 Binary Integers (BINTS)**

Binary integers are the objects you will use most often. They are not the user-level binary integers (those that you enter starting with #); these are actually hexadecimal strings, described in Chapter 7. These system-level binary integers (called bints for short) are objects which are not so easily accessible to the user. If you happen to have one in the stack, they show like part 10h. Try this if you are using a HP49: enter the following number in the stack (triple check if it is right): #3316Bh. Now, type SYSEVAL and press ENTER. You should get part 10h in the stack, or perhaps part 16d (or even something else), depending on the number base you are using. Internally, they are always in hexadecimal mode. With the HP49 and library 256 attached (see Appendix A), you can use the commands R~SB and SB~B to convert reals and user-level binary numbers into bints, respectively, and vice-versa.

Bints are the objects you will use most often because most commands that require a numeric argument need it in the form of a binary integer, as opposed to the real numbers needed by user functions. So, they should be easy to create. And, indeed, they are. You can put one in stack just by entering it on your program (in decimal form). But that is not recommended at all times, because you can also put a real number in stack by just entering it in the same way (we will see later how to differ one from another). So, it is a good idea to use the following structure: # <hex>. This way, you can be sure you will get a binary number, and your code is clearer. Unfortunately (or fortunatelly), you must use hexadecimal representation.

In the HP49G ROM, there are many "built-in" binary numbers. You can put one of these in the stack by just calling its address. Since almost all of them are supported, to get #6h in the stack, you just use the word BINT6. The main advantage is that if you enter # 6, it takes five bytes. The word BINT6, as all other commands (except rompointer and flashpointer commands), take only 2.5 bytes. Some words put two or even three bints in the stack, so the savings are even greater. Following, there is a list of built-in bints.

The four basic operations with bints are #+, #-, #\* and #/. There are

also many others, which are listed below.

Here is an example of program that just put three bints in the stack, using the three methods:

#### 2.1 Reference

#### 2.1.1 Built-in Binary Integers

Addr.	Name	Description
33107	BINT0	0d 0h
		aka: ZERO, any
33111	BINT1	1d 1h
		aka: ONE, real, MEMERR
3311B	BINT2	2d 2h
		aka: TWO, cmp
33125	BINT3	3d 3h
		aka: THREE, str
3312F	BINT4	4d 4h
		aka: FOUR, arry
33139	BINT5	5d 5h
		aka: FIVE, list
33143	BINT6	6d 6h
		aka: SIX, id, idnt
3314D	BINT7	7d 7h
		aka: SEVEN, lam
33157	BINT8	8d 8h
		aka: EIGHT, seco
33161	BINT9	9d 9h
		aka: NINE, symb
3316B	BINT10	10d Ah
		aka: TEN, sym

2.1. Reference

Addr.	Name	Description
33175	BINT11	11d Bh
		aka: ELEVEN, hxs
3317F	BINT12	12d Ch
		aka: TWELVE, grob
33189	BINT13	13d Dh
		aka: TAGGED, THIRTEEN
33193	BINT14	14d Eh
		aka: EXT, FOURTEEN, unitob
3319D	BINT15	15d Fh
		aka: FIFTEEN, rompointer
331A7	BINT16	16d 10h
		aka: REALOB, SIXTEEN
331B1	BINT17	17d 11h
		aka: SEVENTEEN, 2REAL, REALREAL
331BB	BINT18	18d 12h
		aka: EIGHTEEN
331C5	BINT19	19d 13h
		aka: NINETEEN
331CF	BINT20	20d 14h
		aka: TWENTY
331D9	BINT21	21d 15h
		aka: TWENTYONE
331E3	BINT22	22d 16h
		aka: TWENTYTWO
331ED	BINT23	23d 17h
		aka: TWENTYTHREE
331F7	BINT24	24d 18h
		aka: TWENTYFOUR
33201	BINT25	25d 19h
		aka: TWENTYFIVE
3320B	BINT26	26d 1Ah
		aka: REALSYM, TWENTYSIX
33215	BINT27	27d 1Bh
		aka: TWENTYSEVEN
3321F	BINT28	28d 1Ch
		aka: TWENTYEIGHT
33229	BINT29	29d 1Dh
		aka: TWENTYNINE

Addr.	Name	Description
33233	BINT30	30d 1Eh
		aka: REALEXT, THIRTY
3323D	BINT31	31d 1Fh
		aka: THIRTYONE
33247	BINT32	32d 20h
		aka: THIRTYTWO
33251	BINT33	33d 21h
		aka: THIRTYTHREE
3325B	BINT34	34d 22h
		aka: THIRTYFOUR
33265	BINT35	35d 23h
		aka: THIRTYFIVE
3326F	BINT36	36d 24h
		aka: TTHIRTYSIX
33279	BINT37	37d 25h
		aka: THIRTYSEVEN
33283	BINT38	38d 26h
		aka: THIRTYEIGHT
3328D	BINT39	39d 27h
		aka: THIRTYNINE
33297	BINT40	40d 28h
		aka: FORTY, FOURTY
332A1	BINT41	41d 29h
		aka: FORTYONE
332AB	BINT42	42d 2Ah
		aka: FORTYTWO
332B5	BINT43	43d 2Bh
		aka: FORTYTHREE
332BF	BINT44	44d 2Ch
		aka: FORTYFOUR
332C9	BINT45	45d 2Dh
		aka: FORTYFIVE
332D3	BINT46	46d 2Eh
		aka: FORTYSIX
332DD	BINT47	47d 2Fh
	10	aka: FORTYSEVEN
332E7	BINT48	48d 30h
		aka: FORTYEIGHT

Addr.	Name	Description
332F1	BINT49	49d 31h
		aka: FORTYNINE
332FB	BINT50	50d 32h
		aka: FIFTY
33305	BINT51	51d 33h
		aka: FIFTYONE
3330F	BINT52	52d 34h
		aka: FIFTYTWO
33319	BINT53	53d 35h
		aka: FIFTYTHREE, STRLIST, THREEFIVE
33323	BINT54	54d 36h
		aka: FIFTYFOUR
3332D	BINT55	55d 37h
		aka: FIFTYFIVE
33337	BINT56	56d 38h
		aka: FIFTYSIX
33341	BINT57	57d 39h
		aka: FIFTYSEVEN
3334B	BINT58	58d 3Ah
		aka: FIFTYEIGHT
33355	BINT59	59d 3Bh
		aka: FIFTYNINE
3335F	BINT60	60d 3Ch
		aka: SIXTY
33369	BINT61	61d 3Dh
		aka: SIXTYONE
33373	BINT62	62d 3Eh
		aka: SIXTYTWO
3337D	BINT63	63d 3Fh
		aka: SIXTYTHREE
33387	BINT64	64d 40h
		aka: BINT40h, SIXTYFOUR, YHI
33391	BINT65	65d 41h
		aka: ARRYREAL
3339В	BINT66	66d 42h
		aka: FOURTWO
333A5	BINT67	67d 43h
		aka: FOURTHREE

Addr.	Name	Description
333AF	BINT68	68d 44h
		aka: SIXTYEIGHT
333B9	BINT69	69d 45h
		aka: FOURFIVE
333C3	BINT70	70d 46h
		aka: SEVENTY
333CD	BINT71	71d 47h
333D7	BINT72	72d 48h
333E1	BINT73	73d 49h
333EB	BINT74	74d 4Ah
		aka: SEVENTYFOUR
333F5	BINT75	75d 4Bh
333FF	BINT76	76d 4Ch
33409	BINT77	77d 4Dh
33413	BINT78	78d 4Eh
3341D	BINT79	79d 4Fh
		aka: SEVENTYNINE
33427	BINT80	80d 50h
		aka: EIGHTY
33431	BINT81	81d 51h
		aka: EIGHTYONE, LISTREAL
3343B	BINT82	82d 52h
		aka: LISTCMP
33445	BINT83	83d 53h
		aka: FIVETHREE
3344F	BINT84	84d 54h
		aka: FIVEFOUR
33459	BINT85	85d 55h
		aka: 2LIST
33463	BINT86	86d 56h
		aka: FIVESIX
3346D	BINT87	87d 57h
		aka: LISTLAM
33477	BINT88	88d 58h
33481	BINT89	89d 59h
3348B	BINT90	90d 5Ah
33495	BINT91	91d 5Bh
		aka: BINT_91d
3349F	BINT92	92d 5Ch

Addr.	Name	Description
334A9	BINT93	93d 5Dh
334B3	BINT94	94d 5Eh
334BD	BINT95	95d 5Fh
334C7	BINT96	96d 60h
		aka: BINT_96d
334D1	BINT97	97d 61h
		aka: IDREAL
334DB	BINT98	98d 62h
334E5	BINT99	99d 63h
334EF	BINT100	100d 64h
		aka: ONEHUNDRED
334F9	BINT101	101d 65h
33503	BINT102	102d 66h
3350D	BINT103	103d 67h
33517	BINT104	104d 68h
33521	BINT105	105d 69h
3352B	BINT106	106d 6Ah
33535	BINT107	107d 6Bh
3353F	BINT108	108d 6Ch
33549	BINT109	109d 6Dh
33553	BINT110	110d 6Eh
3355D	BINT111	111d 6Fh
		aka: char
33567	BINT112	112d 70h
33571	BINT113	113d 71h
3357B	BINT114	114d 72h
33585	BINT115	115d 73h
		aka: BINT_115d
3358F	BINT116	116d 74h
		aka: BINT_116d
33599	BINT117	117d 75h
335A3	BINT118	118d 76h
335AD	BINT119	119d 77h
335B7	BINT120	120d 78h
335C1	BINT121	121d 79h
335CB	BINT122	122d 7Ah
		aka: BINT_122d
335D5	BINT123	123d 7Bh
335DF	BINT124	124d 7Ch

Addr.	Name	Description
335E9	BINT125	125d 7Dh
335F3	BINT126	126d 7Eh
335FD	BINT127	127d 7Fh
33607	BINT128	128d 80h
		aka: BINT80h
33611	BINT129	129d 81h
3361B	BINT130	130d 82h
		aka: BINT130d, BINT_130d, XHI-1
33625	BINT131	131d 83h
		aka: BINT_131d, BINT131d, XHI
3362F	(#8F)	143d 8Fh
33639	SYMBREAL	145d 91h
33643	(#92)	146d 92h
3364D	(#9A)	154d 9Ah
33657	SYMBUNIT	158d 9Eh
3EAFB	(#9F)	159d 9Fh
3366B	SYMOB	160d A0h
33675	SYMREAL	161d A1h
3367F	(#A2)	162d A2h
39E6B	(#A4)	164d A4h
33689	(#A5)	165d A5h
33693	SYMID	166d A6h
3369D	SYMLAM	167d A7h
336A7	(#A9)	169d A9h
336B1	SYMSYM	170d AAh
336BB	SYMEXT	174d AEh
3BD4C	(#AF)	175d AFh
336C5	(HXSREAL)	177d B1h
38275	(#BB)	187d BBh
336CF	(2HXS)	187d BBh
336D9	BINTC0h	192d C0h
3E7DA	(#C8)	200d C8h
336E3	2GROB	204d CCh
3BD65	(#CF)	207d CFh
336ED	TAGGEDANY	208d D0h
336F7	EXTREAL	225d E1h
33701	EXTSYM	234d EAh
3370B	2EXT	238d EEh
33715	ROMPANY	240d F0h

Addr.	Name	Description
3371F	BINT253	253d FDh
33729	BINT255d	255d FFh
33733	REALOBOB	256d 100h
3373D	#_102	258d 102h
33747	#SyntaxErr	262d 106h
33751	(BINT_263d)	263d 107h
3375B	(#110)	272d 110h
33765	3REAL	273d 111h
3E17B	(#111)	273d 111h
3376F	(Err#Kill)	291d 123h
33779	(Err#NoLstStk)	292d 124h
2777E	(#12F)	303d 12Fh
33783	(#NoRoomForSt)	305d 131h
3378D	(#132)	306d 132h
33797	(REALSTRSTR)	307d 133h
337A1	(#134)	308d 134h
337AB	(#135)	309d 135h
337B5	(#136)	310d 136h
337BF	(#137)	311d 137h
337C9	(#138)	312d 138h
337D3	(#139)	313d 139h
337DD	(#13A)	314d 13Ah
337E7	(#13B)	315d 13Bh
337F1	(#13D)	317d 13Dh
337FB	(#13E)	318d 13Eh
33805	INTEGER337	337d 151h
3380F	(#200)	512d 200h
33819	(Err#NoLstArg)	517d 205h
3A1C2	(#304)	772d 304h
33823	STRREALREAL	785d 311h
3B9FA	(#313)	787d 313h
3C11E	(#410)	1040d 410h
3B928	(#411)	1041d 411h
3382D	(ARRYREALREAL)	1041d 411h
33837	(#412)	1042d 412h
3BA2D	(#414)	1044d 414h
3B93D	(#415)	1045d 415h
33841	(#444)	1092d 444h
3C10F	(#450)	1104d 450h

Addr.	Name	Description
3B952	(#451)	1105d 451h
3384B	(ARRYLISTREAL)	1105d 451h
33855	(#452)	1106d 452h
3BA18	(#454)	1108d 454h
3B913	(#455)	1109d 455h
3A12D	(#4FF)	1279d 4FFh
3385F	(#510)	1296d 510h
33869	(#511)	1297d 511h
3BA09	(#515)	1301d 515h
33873	(#550)	1360d 550h
277F6	(#605)	1541d 605h
27800	(#606)	1542d 606h
2780A	(#607)	1543d 607h
27814	(#608)	1544d 608h
2781E	(#609)	1545d 609h
27828	(#60A)	1546d 60Ah
27832	(#60B)	1547d 60Bh
2783C	(#60C)	1548d 60Ch
27846	(#60D)	1549d 60Dh
2768E	(#60E)	1550d 60Eh
27698	(#60F)	1551d 60Fh
3387D	(IDREALOB)	1552d 610h
276AC	(#611)	1553d 611h
276B6	(#612)	1554d 612h
276C0	(#613)	1555d 613h
276CA	(#614)	1556d 614h
276D4	(#615)	1557d 615h
276DE	(#616)	1558d 616h
276E8	(#617)	1559d 617h
27792	(#618)	1560d 618h
2779C	(#619)	1561d 619h
277A6	(#61A)	1562d 61Ah
277B0	(#61B)	1563d 61Bh
277BA	(#61C)	1564d 61Ch
277C4	(#61D)	1565d 61Dh
277CE	(#61E)	1566d 61Eh
277D8	(#61F)	1567d 61Fh
277E2	(#620)	1568d 620h
277EC	(#621)	1569d 621h

276F2 (#622) 1570d 622h 276FC (#623) 1571d 623h 27706 (#624) 1572d 624h 27710 (#628) 1576d 628h 2771A (#629) 1577d 629h 27724 (#62A) 1578d 62Ah 2772E (#62B) 1579d 62Bh 27738 (#62C) 1580d 62Ch 27742 (#62D) 1581d 62Dh 27788 (#62E) 1582d 62Eh 33887 (IDLISTOB) 1616d 650h 33891 (#700) 1792d 700h 3C17A (#710) 1808d 710h
27706 (#624) 1572d 624h 27710 (#628) 1576d 628h 2771A (#629) 1577d 629h 27724 (#62A) 1578d 62Ah 2772E (#62B) 1579d 62Bh 27738 (#62C) 1580d 62Ch 27742 (#62D) 1581d 62Dh 27788 (#62E) 1582d 62Eh 33887 (IDLISTOB) 1616d 650h 33891 (#700) 1792d 700h
27710       (#628)       1576d 628h         2771A       (#629)       1577d 629h         27724       (#62A)       1578d 62Ah         2772E       (#62B)       1579d 62Bh         27738       (#62C)       1580d 62Ch         27742       (#62D)       1581d 62Dh         27788       (#62E)       1582d 62Eh         33887       (IDLISTOB)       1616d 650h         33891       (#700)       1792d 700h
2771A       (#629)       1577d 629h         27724       (#62A)       1578d 62Ah         2772E       (#62B)       1579d 62Bh         27738       (#62C)       1580d 62Ch         27742       (#62D)       1581d 62Dh         27788       (#62E)       1582d 62Eh         33887       (IDLISTOB)       1616d 650h         33891       (#700)       1792d 700h
27724 (#62A)       1578d 62Ah         2772E (#62B)       1579d 62Bh         27738 (#62C)       1580d 62Ch         27742 (#62D)       1581d 62Dh         27788 (#62E)       1582d 62Eh         33887 (IDLISTOB)       1616d 650h         33891 (#700)       1792d 700h
2772E       (#62B)       1579d 62Bh         27738       (#62C)       1580d 62Ch         27742       (#62D)       1581d 62Dh         27788       (#62E)       1582d 62Eh         33887       (IDLISTOB)       1616d 650h         33891       (#700)       1792d 700h
27738 (#62C)       1580d 62Ch         27742 (#62D)       1581d 62Dh         27788 (#62E)       1582d 62Eh         33887 (IDLISTOB)       1616d 650h         33891 (#700)       1792d 700h
27742 (#62D)       1581d 62Dh         27788 (#62E)       1582d 62Eh         33887 (IDLISTOB)       1616d 650h         33891 (#700)       1792d 700h
27788 (#62E) 1582d 62Eh 33887 (IDLISTOB) 1616d 650h 33891 (#700) 1792d 700h
33887 (IDLISTOB) 1616d 650h 33891 (#700) 1792d 700h
33891 (#700) 1792d 700h
3C17A (#710) 1808d 710h
···· /
3C16B (#750) 1872d 750h
08DF7 (#7FF) 2047d 7FFh
27878 (#800) 2048d 800h
3B976 (#822) 2082d 822h
3C83C (#82C) 2092d 82Ch
3B967 (#855) 2133d 855h
3C81E (#85C) 2140d 85Ch
3389B (#861) 2145d 861h
338A5 (#862) 2146d 862h
338AF (#865) 2149d 865h
338B9 (#86E) 2158d 86Eh
3E7FF (#8F1) 2289d 8F1h
3E759 (#8FD) 2301d 8FDh
3E7E9 (#9F1) $2545d$ 9F1h
3E743 (#9FD) 2557d 9FDh
2774C (#A01) 2561d A01h
27756 (#A02) 2562d A02h
27882 Attn# <b>2563d A03h</b>
338C3 ATTNERR <b>2563d A03h</b>
27760 (#A04) 2564d A04h
2776A (#A05) 2565d A05h
27774 (#A06) <b>2566d A06h</b>
338CD (#A11) 2577d A11h
338D7 (#A12) 2578d A12h
338E1 (#A1A) 2586d A1Ah
338EB (#A21) 2593d A21h

Addr.	Name	Description
338F5	(#A22)	2594d A22h
338FF	(#A2A)	2602d A2Ah
33909	(#A61)	2657d A61h
33913	(#A62)	2658d A62h
3391D	(#A65)	2661d A65h
33927	(#A6E)	2670d A6Eh
33931	(#AA1)	2721d AA1h
3393B	(#AA2)	2722d AA2h
33945	(#AAA)	2730d AAAh
3394F	(#C06)	3078d C06h
33959	(#C07)	3079d C07h
33963	(#C08)	3080d C08h
3396D	Connecting	3082d C0Ah
33977	(#C0B)	3083d C0Bh
3C800	(#C2C)	3116d C2Ch
3C7E2	(#C5C)	3164d C5Ch
3B904	(#C22)	3106d C22h
3B8F5	(#C55)	3157d C55h
33981	#CAlarmErr	3583d DFFh
3398B	EXTOBOB	3584d E00h
3C8D0	(#2111)	8465d 2111h
03FEF	(#2614)	9748d 2614h
03FF9	(#2686)	9862d 2686h
03F8B	TYPEREAL	10547d 2933h
03FDB	(TYPEEREL)	10581d 2955h
03FA9	TYPEIDNT	10568d 2948h
03F95	(TYPECMP)	10615d 2977h
03F9F	(TYPELIST)	10868d 2A74h
20D6F	(TYPERRP)	10902d 2A96h
03FBD	(TYPESYMB)	10936d 2AB8h
03FE5	(TYPEEXT)	10970d 2ADAh
03FA9	(#2E48)	11848d 2E48h
03FD1	(TYPELAM)	11885d 2E6Dh
3C8DF	(#5B11)	23313d 5B11h
3D50D	(#A110)	41232d A110h
3D52B	(#A1A0)	41376 A1A0h
3D51C	(#AA10)	43536d AA10h
2C4D2	(#AAA0)	43680d AAA0h
3B7AD	(#BBBB)	48059d BBBBh

Addr.	Name	Description
08F1F	(#D6A8)	54952d D6A8h
38266	(#FFFF)	65535d FFFFh
03880	(#102A8)	66216d 102A8h
091B4	(#2D541)	185665d 2D541h
350F5	(#37258)	225880d 37258h
0803F	(#414C1)	267457d 414C1h
08ECE	(#536A8)	341672d 536A8h
0657E	(#61441)	398401d 61441h
33995	#EXITERR	458752d 70000h
03826	(#A8241)	688705d A8241h
39277	(#B437D)	738173d B437Dh
038DC	(#E13A8)	922536d E13A8h
3399F	MINUSONE	1048575d FFFFFh

# 2.1.2 Pushing Several BINTs

Addr.	Name	Description
37287	ZEROZERO	( → #0 #0 )
37294	#ZERO#ONE	( $\rightarrow$ #0 #1 )
37305	#ZERO#SEVEN	( $\rightarrow$ #0 #7 )
36B12	ONEONE	( $\rightarrow$ #1 #1 )
		aka: ONEDUP
37315	#ONE#27	( $\rightarrow$ #1 #27d )
37328	#TWO#ONE	( $\rightarrow$ #2 #1 )
3733A	#TWO#TWO	( $\rightarrow$ #2 #2 )
3734A	#TWO#FOUR	( $\rightarrow$ #2 #4 )
3735C	#THREE#FOUR	( $\rightarrow$ #3 #4 )
3736E	#FIVE#FOUR	( $\rightarrow$ #5 #4 )
37380	ZEROZEROZERO	$( \rightarrow #0 #0 #0 )$
37394	ZEROZEROONE	( $\rightarrow$ #0 #0 #1 )
373A8	ZEROZEROTWO	( $\rightarrow$ #0 #0 #2 )
3558C	DROPZERO	$(ob \rightarrow #0)$
355A5	2DROP00	( ob ob $\rightarrow$ #0 #0 )
3596D	DROPONE	( ob $\rightarrow$ #1 )
36AD6	DUPZERO	( ob $\rightarrow$ ob ob $\#0$ )
36AEA	DUPONE	( ob $\rightarrow$ ob ob #1 )
36B26	DUPTWO	( ob $\rightarrow$ ob ob #2 )
36AFE	SWAPONE	( ob ob' $ ightarrow$ ob' ob #1 )

Addr.	Name	Description
35E75	ZEROSWAP	( ob $\rightarrow$ #0 ob )
360BB	ZEROOVER	( ob $ ightarrow$ ob #0 ob )
36568	ZEROFALSE	( $\rightarrow$ #0 F )
35EA2	ONESWAP	( ob $ ightarrow$ #1 ob )
3657C	ONEFALSE	( $ ightarrow$ #1 F )

#### 2.1.3 Conversion

Addr.	Name	Description
262F1	COERCE	( % → # )
35D08	COERCEDUP	( % $\rightarrow$ # # )
35EB6	COERCESWAP	( ob % $\rightarrow$ # ob )
3F481	COERCE 2	( % %' -> # #' )
262EC	%ABSCOERCE	( % → # )
2F244	(COERCE&CKSGN)	( % $ ightarrow$ # flag )
		TRUE if real is greater 0, else FALSE.
2F31F	C%>#	( C% → # #' )
05A03	HXS>#	( hxs $\rightarrow$ # )
2F17E	2HXSLIST?	( $\{ \text{ hxs hxs'} \} \rightarrow \# \#' )$
		Converts list of two hxs to two bints. Gener-
		ates "Bad Argument Value" for invalid input.
05A51	CHR>#	$( chr \rightarrow # )$
0EF006	^Z2BIN	$(Z \rightarrow \#)$
		Convert Z to bint. Returns FFFFF for over-
		flows. Returns 0 for negative numbers.
19D006	^Z>#	$(z \rightarrow \#)$
		Coerces Z to #, overflow error if Z<0 or
		Z>9999. 10000 is used to insure that the
		#*6 can be represented in BCD on a 5 nibbles
		field.
0F0006	^COERCE2Z	( $z2$ $z1$ $\rightarrow$ $\#2$ $\#1$ )
		Converts 2 zints to bints.

## 2.1.4 Arithmetic Functions

Addr.	Name	Description
03DBC	#+	( # #' → #+#' )
03DEF	#1+	( $\#$ $\rightarrow$ $\#+1$ )
03E2D	#2+	$( \# \rightarrow \#+2 )$
355FD	#3+	( # → #+3 )
35602	#4+	( $\# \rightarrow \#+4$ )
35607	#5+	( $\# \rightarrow \#+5$ )
3560C	#6+	( # → #+6 )
35611	#7+	$( \# \rightarrow \#+7 )$
35616	#8+	( # -> #+8 )
3561B	#9+	( # → #+9 )
35620	#10+	( $\#$ $\rightarrow$ $\#+10$ )
3562A	#12+	( $\#$ $\rightarrow$ $\#+12$ )
03DE0	#-	( # #' -> #-#' )
2F13D	(CK#-)	( # #' -> #''')
		If #' is greater than #, returns #0, otherwise re-
		turns #-#'.
03E0E	#1-	$( \# \rightarrow \#-1 )$
03E4E	#2-	$( \# \rightarrow \#-2 )$
355DF	#3-	$( \# \rightarrow \#-3 )$
355DA	#4-	$( \# \rightarrow \#-4 )$
355D5	#5-	$( \# \rightarrow \#-5 )$
355D0	#6-	$( \# \rightarrow \#-6 )$
03EC2	#*	( # #' -> #*#' )
2632D	#*OVF	( # #' → #*#' )
		$0 \leq \mathrm{result} \leq \mathrm{FFFFF}$
03E6F	#2*	( # → #*2 )
356B8	#6*	( # → #*6 )
3569B	#8*	( # → #*8 )
35675	#10*	( # → #*10 )
03EF7	#/	$( \# \#' \rightarrow \#r \#q )$
03E8E	#2/	$(\# \rightarrow \#/2)$
		Rounded down.
36815	#1-	( # #' → #-#'+1 )
0.60=-		aka: #-+1
36851	#1-+	$(# #' \rightarrow #+#'-1)$
		\$1-+ is a typo in EXTABLE.
		aka: #+-1, \$1-+

```
Addr.
         Name
                             Description
35552
         #-#2/
                             ( # #' \rightarrow (#-#')/2 )
                             ( \# \#' \rightarrow \# + \#' \# + \#' )
357FC
         #+DUP
35E39
                             ( ob # #' \rightarrow #+#' ob )
         #+SWAP
         #+OVER
36093
                             ( ob # #' \rightarrow ob #+#' ob )
3581F
         #-DUP
                             ( \# \#' \rightarrow \#-\#' \#-\#' )
35E4D
         #-SWAP
                             ( ob # #' \rightarrow #-#' ob )
360A7
         #-OVER
                             ( ob \# \#' \rightarrow ob \#-\#' ob )
                             ( \# \rightarrow \#+1 \#+1 )
35830
         #1+DUP
                             ( ob \# \rightarrow \#+1 ob )
35E61
         #1+SWAP
                             ( ob ob' \# \rightarrow ob' \#+1 ob )
2F222 #1+ROT
                             ( \# \rightarrow \#-1 \#-1 )
35841
         #1-DUP
28071
         #1-SWAP
                             ( ob # \rightarrow #-1 ob )
                             aka: pull
3601B
         #1-ROT
                             ( ob ob' \# \rightarrow ob' \#-1 ob )
281D5
         #1-UNROT
                             ( ob ob' \# \rightarrow \#-1 ob ob' )
35E89
         #1-1SWAP
                             ( \# \rightarrow 1 \# -1 )
                             Returns the bint ONE and the result.
                             ( \# \rightarrow \# \#+1 )
35912
         DUP#1+
3571E
         DUP#2+
                             ( \# \rightarrow \# \# + 2 )
35956
         DUP#1-
                             ( \# \rightarrow \# \#-1 )
                             ( \# \#' \rightarrow \# \#' \# + \#' )
3674D
         2DUP#+
                             aka: DUP3PICK#+
                             ( \# ob \rightarrow \#-1 )
3683D
        DROP#1-
357BB
         SWAP#-
                             ( # #' \rightarrow #'-# )
                             ( \# ob \rightarrow ob \#+1 )
3592B
         SWAP#1+
                             aka: SWP1+
29786
         ('RSWAP#1+)
                             ( \# \rightarrow \text{nob } \#+1 )
                             nob is the next object in the runstream.
28099
         SWAP#1+SWAP
                             ( \# ob \rightarrow \#+1 ob )
         SWAP#1-
36829
                             ( \# ob \rightarrow ob \#-1 )
280AD
                             ( \# ob \rightarrow \#-1 ob )
         SWAP#1-SWAP
28989
         (SWAPDROP#1-)
                             ( ob # \to #-1 )
                             ( \# \#' \rightarrow \#' \#-\#' )
367ED
         SWAPOVER#-
                             ( \# \#' \to \# \#' + \# )
36775
         OVER#+
367C5
                             ( # #' \rightarrow # #'-# )
         OVER#-
                             ( # ob #' \rightarrow ob #'+# )
36761
        ROT#+
367B1
        ROT#-
                             ( \# ob \#' \to ob \#'-\# )
                             ( \# ob ob' \rightarrow ob ob' \#+1 )
36801
        ROT#1+
                             ( \# ob ob' \rightarrow \#+1 ob ob' )
28001
        ROT#1+UNROT
```

Addr.	Name	Description
35E07	ROT#+SWAP	( # ob #' → #'+# ob )
		aka: ROT+SWAP
36789	3PICK#+	( $\#$ ob $\#'$ $\rightarrow$ $\#$ ob $\#'$ + $\#$ )
3679D	4PICK#+	( $\#$ ob1 ob2 $\#'$ $\rightarrow$ $\#$ ob1 ob2 $\#'$ + $\#$ )
35E20	4PICK#+SWAP	( $\#$ ob1 ob2 $\#'$ $\rightarrow$ $\#$ ob1 $\#'$ + $\#$ ob2 )
		aka: 4PICK+SWAP
35511	#MIN	( # #' -> #'' )
3551D	#MAX	( # #' -> #'' )
03EB1	#AND	( # #' -> #'' )
		Bitwise AND.

#### 2.1.5 Tests

```
Addr.
          Name
                        Description
03D19
          #=
                        ( \# \rightarrow flaq )
          #<>
                        ( \# \rightarrow flag )
03D4E
03CE4
                        ( \# \rightarrow flag )
          #<
03D83
          #>
                        ( \# \rightarrow flag )
03CC7
          #0<>
                        ( \# \rightarrow flag )
03CA6
          #0=
                        ( \# \rightarrow flag )
3530D
                        ( \# \rightarrow flag )
          #1<>
                        ( \# \rightarrow flag )
352FE
          #1=
36711
          #2<>
                        ( \# \rightarrow flag )
352F1
          #2=
                        ( \# \rightarrow flag )
352E0
          #3=
                        ( \# \rightarrow flag )
366FD
          #5=
                        ( \# \rightarrow flag )
                        ( \# \rightarrow flag )
366BC
          #<3
36739
          \#>1
                        ( \# \rightarrow flag )
                        aka: ONE#>
                        ( \# \#' \rightarrow \# \#' flag )
358C2
          2DUP#<
                        ( \# \#' \rightarrow \# \#' flag )
358F8
          2DUP#>
                        ( \# \rightarrow flag )
363CE
          ONE_EQ
                        Uses EQ test.
35268
         OVER#=
                        ( \# \#' \rightarrow \# flag )
                        ( \# \#' \rightarrow \# \#' flag )
358DC
          2DUP#=
36694
          OVER#0=
                        ( \# \#' \rightarrow \# \#' flag )
                        ( \# \rightarrow \# flag )
352BD
          DUP#0=
                        ( \# \#' \rightarrow \# flag )
366A8
         OVER#<
```

Addr.	Name	Description
3531C	DUP#1=	( $\# \rightarrow \#$ flag )
36725	OVER#>	( $\#$ $\#'$ $\to$ $\#$ flag )
3532B	DUP#0<>	( $\# \rightarrow \#$ flag )
366D0	DUP#<7	( $\# \to \#$ flag )
		Returns TRUE if the argument is smaller then #7.
36676	2#0=OR	( $\#$ $\#$ $\to$ flag )
		Returns TRUE if either argument is zero.

# Chapter 3 Real Numbers

Real numbers can be created in two ways. The first is by just entering them, without any prefix. But this method can also be used to create bints. So how does the compiler know when you want a real number and when you want a bint? If the number includes a radix and/or an exponent, then it is a real number; otherwise it is a bint.

Because of the possible confusion, the preferred method is to use the structure % <dec>. This way, you will surely get a real number, and the code becomes more readable.

As for bints, there are also many built-in real numbers. They are listed below.

The basic operations using real numbers are +, -, \*\*, and  $^\circ$ . But there are many others, which are listed below.

There is also another kind of real number, which is not directly accessible to the user and to User RPL programs. They are the Extended (or Long) Real Numbers. They work like normal real numbers, with two differences: they have a 15-digit precision opposed to the 12-digit of the normal real numbers, and their exponents are in the range from -50000 to 50000.

Extended real numbers are created using <code>%% <dec></code>. If you happen to get one in the stack, they display like normal reals, but always in scientific notation. The basic operations are the same, except that they are prefixed with <code>%%</code> instead of <code>%</code>. Let me make thing one clear, if it is not already: in User RPL, <code>+</code> adds any kind of objects, for example real numbers, user binary integers (which are hexadecimal strings, as we will see later), adds elements to lists, etc. In System RPL, the word <code>%+</code> only works for two real numbers. To add two binary integers, you must use <code>#+</code>. To add extended reals, the word is <code>%\*+</code>. If you call a function with the wrong arguments, there is a possibility that your system will crash.

To convert from a real number to an extended real number, you can use the command %>%%. The opposite function is %%>%. To convert from a bint to a (normal) real number, the function is UNCOERCE, and the opposite function is 28 3. Real Numbers

COERCE. Below there is a list of more conversion functions, and other functions related to real numbers.

## 3.1 Reference

#### 3.1.1 Built-in Real Numbers

Addr.	Name	Description
2FB0A	%-MAXREAL	-9.99E499
2FAB1	% <b>-</b> 9	-9
2FA9C	%-8	-8
2FA87	%-7	-7
2FA72	%-6	-6
2FA5D	%-5	-5
2FA48	%-4	-4
2FA33	%-3	-3
2FA1E	%−2	-2
2FA09	%-1	-1
2FB34	%-MINREAL	-1E-499
2F937	응0	0
2FB1F	%MINREAL	1E-499
27118	%.1	.1
339BE		.5
339D3	(%5)	5
2F94C	%1	1
270EE	(%1.8)	1.8
2F961	%2	2
339A9	%e	e
2F976	%3	3
2FAC6	%PI	$\pi$
2F98B	<b>%4</b>	4
2F9A0	%5	5
2F9B5		6
2F9CA	%7	7
2F9DF	%8	8
2F9F4	89	9
339E8	%10	10
2FCE6	%11	11

Addr.	Name	Description
2FCFB	%12	12
2FD10	%13	13
2FD25	%14	14
2FD3A	%15	15
2FD4F	%16	16
2FD64	%17	17
2FD79	%18	18
2FD8E	<b>%19</b>	19
2FDA3	%20	20
2FDB8	%21	21
2FDCD	%22	22
2FDE2	%23	23
2FDF7	%24	24
2FE0C	%25	25
2FE21	%26	26
2FE36	%27	27
2FE4B	(%28)	28
2FE60	(%29)	29
2FE75	(%30)	30
2FE8A	(%31)	31
2FE9F	(%32)	32
2FEB4	(%33)	33
2FEC9	(%34)	34
2FEDE	(%35)	35
27103	%80	80
27E5D	%100	100
339FD	%180	180
33A12	(%200)	200
33A3C	(%400)	400
33A27	%360	360
2FC7D	(%1200)	1200
2FC92	(%2400)	2400
2FCA7	(%4800)	4800
4EA22	(%TICKSsec)	8192
2FCBC	(%9600)	9600
2FCD1	(%15360)	15360
4EA37	(%TICKSmin)	491520
4EA4C	(%TICKShour)	29491200
4EA61	(%TICKSday)	707788800

30 3. Real Numbers

Addr.	Name	Description
4EA76	(%TICKSweek)	4954521600
2FAF5	%MAXREAL	9.99E499
2F180	1REV	$( \rightarrow 6.28318530718 )$
		$(\rightarrow 360.)$
		$( \rightarrow 400. )$
		Returns the angle of a full circle, corresponding
		to the current angular mode.

#### 3.1.2 Built-in Extended Real Numbers

Addr.	Name	Description
2FB49	%%O	0
2FBE5	%%.1	0.1
30DC8	%%.4	0.4
2FBFF	%%.5	0.5
2DA11	cfF	0.555
		$\%\%5/9$ for C $\leftrightarrow$ F conversion.
2FB63	%%1	1
2DA2B	cfC	1
		For $C \leftrightarrow K$ conversion.
2FB7D	%%2	2
2FB97	%%3	3
2FADB	%%PI	$\pi$
30017	PI/180	$\pi/180$
2FBB1	884	4
2FBCB	%%5	5
27A89	%%2PI	$2\pi$
30BEA	%%7	7
2FC19	%%10	10
30CC7	%%12	12
30CEB	%%60	60

# 3.1.3 Stack Manipulation Combined with Reals

Addr.	Name	Description
282CC	(DROP%0)	$(ob \rightarrow %0)$

#### 3.1.4 Conversion

Addr.	Name	Description
2FFAC	%>%%	( % → %% )
35ECA	%>%%SWAP	( ob % $\rightarrow$ %% ob )
2FF9B	%%>%	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
30E47	2%>%%	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
30E5B	2%%>%	( %% %%' → % %' )
262F6	UNCOERCE	( ♯ → % )
3F495	UNCOERCE2	( # # -> % % )
36BFA	UNCOERCE%%	( ♯ → %% )
2EFCA	HXS>%	( hxs $\rightarrow$ % )
05D2C	C%>%	( C% $\rightarrow$ %re %im )
2B3FD	%IP>#	$( % \rightarrow \#IP(ABS(%)) )$
		Does ABS too.
0F6006	^Z>R	$(Z \rightarrow %)$
		Converts zint to real.
18A006	^Z2%%	( Z → %% )
		Converts integer to long real.
197006	^OBJ2REAL	( z/% → % )
		Transforms ob in real.

#### 3.1.5 Real Functions

Addr.	Name	Description
3035F	%+	( % %' → %+%' )
25E69	%+SWAP	( ob % %' $\rightarrow$ %+%' ob )
26F36	%1+	$( \% \rightarrow \%+1 )$
3036C	%-	( % %' → %-%')
26F4A	%1-	( % → %-1 )
30346	%>%%-	( % %' → %%-%%')
303A7	%*	( % %' → %*%' )

32 3. Real Numbers

```
Addr.
          Name
                            Description
35C18
          %10*
                               % \rightarrow %*10)
                            ( % %' → %/%' )
303E9
          왕/
3045B
                            ( \% \%' \rightarrow \%^{\%}' )
                            (\% \rightarrow \%')
302EB
          %ABS
                            ( % → -% )
3030B
          %CHS
                               % \rightarrow -1/0/1)
302C2
         %SGN
                            (% \rightarrow 1/%)
3049A
         %1/
30489
         %>%%1/
                            ( \% \rightarrow 1/\% \% )
                               % → √% )
304F4
          %SQRT
                                     ,/응용 )
304E1
          %>%%SQRT
                            ( % →
                               % \rightarrow e^{*}
3051A
          %EXP
3052D
          %EXPM1
                            ( % \rightarrow e^{-1} )
                            ( % \rightarrow LN% )
30559
          %LN
                            ( % \rightarrow LN(%+1) )
30592
          %LNP1
3056C
          %LOG
                            ( % \rightarrow LOG% )
                            ( % \rightarrow 10^{\circ} % )
305A5
          %ALOG
305DA
         %SIN
                            ( % \rightarrow SIN% )
                            ( % \rightarrow COS% )
3062B
         %COS
3067C
          %TAN
                            ( % \rightarrow TAN% )
                            ( % \rightarrow ASIN% )
306AC
          %ASIN
                            ( % \rightarrow ACOS% )
306DC
          %ACOS
                            ( % \rightarrow ATAN% )
3070C
          %ATAN
30799
                            ( % \rightarrow SINH% )
          %SINH
                            ( % \rightarrow COSH% )
307C5
          %COSH
307D8
          %TANH
                            ( % \rightarrow TANH% )
                            ( % \rightarrow ASINH% )
307EB
          %ASINH
                            ( % \rightarrow ACOSH% )
307FE
          %ACOSH
                               % \rightarrow ATANH%)
30811
          %ATANH
                            ( % \rightarrow %mant )
3031B
          %MANTISSA
30824
          %EXPONENT
                            ( % \rightarrow %expn )
                            ( % \rightarrow %frac )
30938
         %FP
3094B
         %IP
                               % \rightarrow % int )
30971
          %FLOOR
                               % \rightarrow % \max (=%)
                            ( % \rightarrow %minint >=% )
3095E
         %CEIL
                               % %' → %rem )
305C7
          %MOD
30723
          %ANGLE
                            ( x y \rightarrow ang )
                            ( x + y \rightarrow x ang )
30746
          %>%%ANGLE
         RNDXY
                            ( % \text{ *places} \rightarrow %' )
30F14
```

Addr.	Name	Description
30F28	TRCXY	( % %places → %' )
3084D	%COMB	$( % %' \rightarrow COMB(%,%') )$
30860	%PERM	( % %' $\rightarrow$ PERM(%,%') )
30837	%NFACT	( % → %! )
		Calculates factorial of number.
30AAF	%FACT	$( % \rightarrow gamma(%+1) )$
		Calculates gamma(x+1).
3046C	%NROOT	$( % %n \rightarrow %' )$
		Calculates the %nth root of the real number.
		Equivalent to user function XROOT.
300F9	%MIN	( % %' $ ightarrow$ %lesser )
300E0	%MAX	$( % %' \rightarrow %greater )$
35DBC	%MAXorder	( % %' $\rightarrow$ %max %min )
309AD	%RAN	$( \rightarrow {random})$
		Returns next random number.
30A2F	%RANDOMIZE	$( \$seed \rightarrow )$
		System level RDZ: seeds the random number gen-
		erator.
30A66	DORANDOMIZE	$( \ \ \ \ \ \ \ \ \ \ \ )$
		Stores given number as random number seed.
303B4	%OF	( % %' → %'/% * 100 )
303F6	%T	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
3041B	%CH	( % %' $ ightarrow$ %pcchange )
3000D	%D>R	( $%deg \rightarrow %rad$ )
30040		$( rad \rightarrow deg )$
	%REC>%POL	( %r %ang → %x %y )
	%POL>%REC	$( %x %y \rightarrow %r %ang )$
30EDD	%SPH>%REC	( %r %ang %ph $ ightarrow$ %x %y %z )

## 3.1.6 Extended Real Functions

Addr.	Name	Description
3032E	응응+	( %% %%' → %%+%%' )
3033A	% % <b>-</b>	( %% %%¹ → %%-%%¹ )
30385	응응*	( %% %%' → %%*%%' )
3602F	%%*ROT	( ob ob' %% %%' $ ightarrow$ ob' %%+%%' ob )
35EDE	%%*SWAP	$( ob  \%  \%  \%  ) \rightarrow  \% + \%        ) $
36C7C	%%*UNROT	( ob ob' %% %%' $\rightarrow$ %%+%%' ob ob' )

3. Real Numbers

```
Addr.
          Name
                           Description
303D3
                           ( \% \% \% ' \rightarrow \% \% / \% \% ' )
           응왕/
                           ( %% %%' \rightarrow %%'' )
36C22
           SWAP%%/
36BE6
           ( \% \% \% \rightarrow \% )
           응응^
                           ( \% \% \% ) \rightarrow \% \% \% )
3044A
51D006
           ^CK%%SQRT
                           ( \% \to \%/\%/\% )
30612
           %%SINRAD
                           ( %% → %%' )
           %%ANGLERAD ( %% \rightarrow %%' )
30767
302DB
                           ( % \rightarrow %abs )
           %%ABS
                           ( % \rightarrow %rad )
306F3
           %%ACOSRAD
                           ( %x %%y \rightarrow %anq )
3073A
           %%ANGLE
           \mbox{\$\$ANGLEDEG} ( \mbox{\$\$x} \mbox{\$\$y} \rightarrow \mbox{\$\$deg} )
30757
                           ( %% \rightarrow %%rad )
306C3
           %%ASINRAD
                           ( %% → -%% )
302FB
           %%CHS
                           ( \% \rightarrow 1/\% )
3047D
           %%1/
30642
                           ( %% \rightarrow %%cos )
           %%COS
                           ( \%deg \rightarrow \%cos )
30653
           %%COSDEG
                           ( % \rightarrow %cosh )
307B2
           %%COSH
                           ( %rad \rightarrow %cos )
30663
           %%COSRAD
30507
           %%EXP
                           ( \% \to e^{\%} )
                           ( %% → ln %% )
30546
           %%LN
                           ( \% \rightarrow \%maxint )
30984
           %%FLOOR
                           aka: %%INT
3057F
           %%LNP1
                           ( \% \rightarrow \% \ln(\% + 1) )
300C7
           %%MAX
                           ( %% %%' \rightarrow %%max )
                           ( %x %%y \rightarrow %radius %angle )
30E83
           %%R>P
30EB0
           %%P>R
                           ( %r %%ang \rightarrow %%x %%y )
                           ( \% \rightarrow \%sin )
305F1
           %%SIN
                           ( \% deg \rightarrow \% sin )
30602
           %%SINDEG
30780
           %%SINH
                           ( \% \rightarrow \%sinh )
                           ( %% → √%% )
304D5
           %%SQRT
                           ( %rad \rightarrow %tan )
30693
           %%TANRAD
```

#### 3.1.7 Tests

```
Addr.
         Name
                     Description
302AC
                     ( % %' \rightarrow flag )
302B7
                     ( % %' \rightarrow flag )
         %<>
3025C
         %<
                     ( % %' \rightarrow flag )
302A1
                     ( % %' \rightarrow flag )
         %<=
30275
                     ( % %' \rightarrow flag )
         %>
                     ( % %' \rightarrow flag )
3028B
         %>=
30156
         응0=
                     ( % \rightarrow flag )
36C0E
         DUP%0=
                    ( % \rightarrow flag )
301BA
         %0<>
                     ( % \rightarrow flag )
                     Can be used to change a user flag into a system flag.
30123
         %0<
                     ( % \rightarrow flag )
30184
                     ( % \rightarrow flag )
         %0>
301E2
         %0>=
                     ( % \rightarrow flag )
3020A
                     ( %% %%' \rightarrow flag )
         %왕<
                     ( %% %%' \rightarrow falg )
30296
         %%<=
3026A
         %%>
                     ( %% %%' \rightarrow flag )
30280
                     ( %% %%' \rightarrow flag )
         %%>=
30145
         응응0=
                     ( % \rightarrow flag )
                     ( % \rightarrow flag )
301A6
         %%0<>
                     ( % \rightarrow flag )
30112
         %%0<
301F6
         %%0<=
                     ( \% \rightarrow flaq )
                     ( % \rightarrow flag )
30173
         %%0>
                     ( %% \rightarrow flag )
301CE
        %%0>=
```

# Chapter 4 Complex Numbers

Complex numbers can be inserted in your program with the following structure: C% <real> <imag>. The real and imaginary parts are real numbers, in decimal form. If you have the real and imaginary parts in the stack, the command %>C% will create a complex number from them. The command C%>% takes a complex number and returns the real and imaginary parts.

There exists also the Extended (also called Long) Complex Numbers, which are not directly accessible to the user. They are complex number whose real and imaginary parts are extended reals. They can be inserted in your program with C%% <real> <imag>, where the real and imaginary parts are extended reals. They show in the stack as a normal complex number, but always in scientific notation.

Below is a list of all the commands related to complex numbers, including mathematical operations.

#### 4.1 Reference

#### 4.1.1 Builtin Complex Numbers

Addr.	Name	Description
27DE4	C%0	(0,0)
27E09	C%1	(1,0)
27DBF	C%-1	(-1,0)
27E2E	C%%1	(%%1,%%0)

## 4.1.2 Conversion

Addr.	Name	Description
261D9	C%%>C%	( C%% → C% )
05C27	%>C%	( %re %im $ ightarrow$ C% )
362F2	SWAP%>C%	( %im %re $ ightarrow$ C% )
261FC	Re>C%	( %re $ ightarrow$ C% )
25E9C	C>Re%	( C% $\rightarrow$ %re )
25E9B	C>Im%	( C% $\rightarrow$ %im )
18C006	^E%%>C%%	( %%re %%im $\rightarrow$ C%% )
		Converts long reals to long complex.
261CF	%%>C%	( %%re %%im $\rightarrow$ C% )
25E82	C%>%%	( C% $\rightarrow$ %%re %%im )
25E83	C%>%%SWAP	( C% $\rightarrow$ %%im %%re )
05DBC	C%%>%%	( C%% $\rightarrow$ %%re %%im )
188006	^C2C%%	( C → C%% )
		Converts Gaussian integer to long complex.
189006	^ZZ2C%%ext	( Zre Zim $\rightarrow$ C%% )
		Converts Gaussian integer to long complex.
18B006	^C%>C%%	( C% → C%% )
		Converts complex to long complex.
15E006	^RIXCext	( Zre Zim $ ightarrow$ C )
		Convert integers to complex.
15F006	^IRXCext	( Zim Zre $ ightarrow$ C )
		Convert integers to complex.

## 4.1.3 Functions

Addr.	Name	Description
25E8F	C%C^C	( C% C%' → C%'' )
25E90	C%C^R	( C% % → C%' )
25E94	C%R^C	( % C% → C%' )
25E84	C%ABS	( C% → % )
50C006	^CZABS	( complex $ ightarrow$ real )
		Absolute value.
261ED	C%CHS	( C% → -C% )
25E81	C%1/	( C% → 1/C% )
25E98	C%SQRT	( C% → √C% )
25E95	C%SGN	( $C\% \rightarrow C\%/C\%ABS$ )

Addr.	Name	Description
261F2	C%CONJ	( C% → C%' )
25E88	C%ARG	( C% → % )
25E91	C%EXP	$(C% \rightarrow e^{C}%)$
25E92	C%LN	( $C\% \rightarrow ln C\%$ )
25E93	C%LOG	( C% $ ightarrow$ log C% )
25E87	C%ALOG	( $C\% \rightarrow 10^{C}\%$ )
25E96	C%SIN	( $C\% \rightarrow \sin C\%$ )
25E8D	C%COS	( $C\% \rightarrow \cos C\%$ )
25E99	C%TAN	( $C\% \rightarrow tan C\%$ )
25E89	C%ASIN	( $C\% \rightarrow asin C\%$ )
25E85	C%ACOS	( $C\% \rightarrow acos C\%$ )
25E8B	C%ATAN	( $C\% \rightarrow atan \ C\%$ )
25E97	C%SINH	( $C\% \rightarrow sinh C\%$ )
25E8E	C%COSH	( $C\% \rightarrow cosh C\%$ )
25E9A	C%TANH	( $C\% \rightarrow tanh \ C\%$ )
25E8A	C%ASINH	( $C\% \rightarrow asinh C\%$ )
25E86	C%ACOSH	$(C\% \rightarrow acosh C\%)$
25E8C	C%ATANH	( $C\% \rightarrow atanh \ C\%$ )
261DE	C%%CHS	( C%% → -C%% )
261E3	C%%CONJ	( C%% → C%%' )
515006	^ARG2	( im re $\rightarrow$ arg(ob) )
		ARG.
517006	^QUADRANT	( re im $?re>0$ $?im>0 \rightarrow newre newim % )$
		Returns Z0 Z1 Z-2 or Z-1 so that arg of correspond-
		ing complex number is $Z * \pi/2$ + theta where $\theta$ is in
F1006	^ 0 0 0 000 00	the interval $[0,\pi/2]$ .
51E006	^C%%SQRT	( C%% → C%%' )

#### 4.1.4 Tests

# Addr. Name Description 261E8 C%0= ( C% → flag )

261D4 C%%0= ( C%%  $\rightarrow$  flag )

# **Chapter 5 Integers (ZINTS)**

This is a new object of the HP49. The integers (called ZINT's for shorts) are a numerical type that can represent arbitrarily large integers.

In most cases, you do not really need to worry about integers entered by the user as arguments for a program. The type checking mechanism (described in section 29.2) will in most cases transparently convert zints to real numbers.

If you want to work with integers, however, there are several functions dealing with zints. Since this object type is really a part of the HP49 CAS, these functions are not described here. Instead, turn to Chapter 42 for documentation on ZINTs.

# **Chapter 6 Characters and Strings**

Characters and strings are two data types that hold text.

Characters are not directly available to the user. They can only hold one character. You create them with CHR <char> or using one of the many built-in characters (listed below). To convert a character to a bint, use CHR>#. The bint returned is the ASCII code for the character. The opposite function is #>CHR.

Strings are inserted in your program with \$ "<string>", or simply "<string>". There are some built-in strings, listed below. It is possible to convert a character into a string, with the command CHR>\$.

Two useful and simple functions which deal with strings are LEN\$ and &\$. The first returns the length (the number of characters) of a string as a bint, and the second concatenates two strings. To get a substring, i.e., part of a string, use the function SUB\$. It expects three arguments: the original string, the starting position (a bint) and the final position (also a bint). Counting starts at one. Everything between the start and end characters (inclusive) will be returned. And another function is POS\$, which searches a string (in level three) for a character or string (in level two), starting from a specified position (a bint, in level one). The position of the first occurrence of the search string in the string is returned (as a bint) to level one. If it could not be found, #0 is returned. There are also many other functions, see below for a list.

# 6.1 Reference

# 6.1.1 Built-in Characters

Addr.	Name	Desc	eription				
33D2B	CHR_00	'\00'	(character	0d	00h)		
		The 2	NULL charae	cter.			
33F77	CHR_Newline	'\0a'	(character	10d	0Ah)		
		The	newline char	acter.			
33D32	CHR	<b>'</b> '	(character	31d	1Fh)		
33F93	CHR_Space	' '	(character	32d	20h)		
		The s	space charac	ter.			
33D39	CHR_DblQuote	1111	(character	34d	22h)		
33D40	CHR_#	'#'	(character	35d	23h)		
33F70	CHR_LeftPar	'('	(character	40d	28h)		
33F85	CHR_RightPar	')'	(character	41d	29h)		
33D47	CHR_*	'*'	(character	42d	2Ah)		
33D4E	CHR_+	'+'	(character	43d	2Bh)		
33D55	CHR_,	','	(character	44d	2Ch)		
33D5C	CHR	'-'	(character	45d	2Dh)		
33D63	CHR	'.'	(character	46d	2Eh)		
33D6A	CHR_/	'/'	(character	47d	2Fh)		
33D71	CHR_0	'0'	(character	48d	30h)		
33D78	CHR_1	'1'	(character	49d	31h)		
33D7F	CHR_2	'2'	(character	50d	32h)		
33D86	CHR_3	'3'	(character	51d	33h)		
33D8D	CHR_4	'4'	(character	52d	34h)		
33D94	CHR_5	'5'	(character	53d	35h)		
33D9B	CHR_6	'6'	(character	54d	36h)		
33DA2	CHR_7	'7'	(character	55d	37h)		
33DA9	CHR_8	'8'	(character	56d	38h)		
33DB0	CHR_9	'9'	(character	57d	39h)		
33DB7	CHR_:	<b>':'</b>	(character		3 <b>A</b> h)		
33DBE	CHR_;	'.' ,	(character		3Bh)		
33DC5	CHR_<	'<'	(character				
33DCC	CHR_=	'='	(character				
33DD3	CHR_>	'>'	(character		3Eh)		
33DDA	CHR_A	'A'	(character		41h)		
33DE1	CHR_B	'B'	(character	66d	42h)		

Addr.	Name	Desc	cription			
33DE8	CHR_C	'C'	(character	67d	43h)	
33DEF	CHR_D	D'	(character	68d	44h)	
33DF6	CHR_E	'E'	(character	69d	45h)	
33DFD	CHR_F	'F'	(character	70d	46h)	
33E04	CHR_G	'G'	(character	71d	47h)	
33E0B	CHR_H	Ή'	(character	72d	48h)	
33E12	CHR_I	Ί'	(character	73d	49h)	
33E19	CHR_J	'J'	(character	74d	4Ah)	
33E20	CHR_K	'K'	(character	75d	4Bh)	
33E27	CHR_L	$^{\prime}\mathrm{L}^{\prime}$	(character	76d	4Ch)	
33E2E	CHR_M	'M'	(character	77d	4Dh)	
33E35	CHR_N	'N'	(character	78d	4Eh)	
33E3C	CHR_O	'O'	(character	79d	4Fh)	
33E43	CHR_P	'P'	(character	80d	50h)	
33E4A	CHR_Q	'Q'	(character	81d	51h)	
33E51	CHR_R	'R'	(character	82d	52h)	
33E58	CHR_S	'S'	(character	83d	53h)	
33E5F	CHR_T	'T'	(character	84d	54h)	
33E66	CHR_U	'U'	(character	85d	55h)	
33E6D	CHR_V	'V'	(character	86d	56h)	
33E74	CHR_W	'W'	(character	87d	57h)	
33E7B	CHR_X	'X'	(character	88d	58h)	
33E82	CHR_Y	'Y'	(character		59h)	
33E89	CHR_Z	'Z'	(character	90d	5Ah)	
33FA1	CHR_[	Έ'	(character		5Bh)	
33FA8	CHR_]	']'	(character		5Dh)	
33F9A	CHR_UndScore	'_'	(character	95d	5Fh)	
33E90	CHR_a	'a'	(character		61h)	
33E97	CHR_b	'b'	(character		62h)	
33E9E	CHR_c	'c'	(character		63h)	
33EA5	CHR_d	'd'	(character	100d	64h)	
33EAC	CHR_e	'e'	(character			
33EB3	CHR_f	'f'	(character	102d	66h)	
33EBA	CHR_g	'g'	(character			
33EC1	CHR_h	'h'	(character	104d	68h)	
33EC8	CHR_i	'i'	(character			
33ECF	CHR_j	'j'	(character	106d	6Ah)	
33ED6	CHR_k	'k'	(character			
33EDD	CHR_1	'l'	(character	108d	6Ch)	

Addr.	Name	Des	cription
33EE4	CHR_m	'm'	(character 109d 5Dh)
33EEB	CHR_n	'n'	(character 110d 6Eh)
33EF2	CHR_o	'o'	(character 111d 6Fh)
33EF9	CHR_p	'p'	(character 112d 70h)
33F00	CHR_q	$'\mathbf{q}'$	(character 113d 71h)
33F07	CHR_r	'r'	(character 114d 72h)
33F0E	CHR_s	's'	(character 115d 73h)
33F15	CHR_t	't'	(character 116d 74h)
33F1C	CHR_u	'u'	(character 117d 75h)
33F23	CHR_v	'v'	(character 118d 76h)
33F2A	CHR_w	'w'	(character 119d 77h)
33F31	CHR_x	'x'	(character 120d 78h)
33F38	CHR_y	'y'	(character 121d 79h)
33F3F	CHR_z	$\mathbf{z}'$	(character 122d 7Ah)
33FAF	$CHR_{$	'{'	(character 123d 7Bh)
33FB6	$CHR_{}$	'{'	(character 125d 7Dh)
33F5B	CHR_Angle	<b>'</b> ∠'	(character 128d 80h)
33F69	CHR_Integral	'∫'	(character 132d 84h)
33F62	CHR_Deriv	'∂'	(character 136d 88h)
33F46	CHR>	<b>'</b> →'	(character 141d 8Dh)
33F4D	CHR_<<	'«'	(character 171d ABh)
33F54	CHR_>>	'»'	(character 187d BBh)
33F7E	CHR_Pi	$'\pi'$	(character 135d 87h)
33F8C	CHR_Sigma	$\Sigma'$	(character 133d 85h)
33FBD	CHR_<=	'≤' '≥'	(character 137d 89h)
33FC4	CHR_>=		(character 138d 8Ah)
33FCB	CHR_<>	'≠'	(character 139d 8Bh)

# 6.1.2 Built-in Strings

Addr.	Name	Description
055DF	NULL\$	IIII
		Empty string.
33B55	SPACE\$	" " 
		aka: tok_
33B39	NEWLINE\$	"\0a"
		Newline.

Addr.	Name	Description
27195	CRLF\$	"\0d\0a"
		Carriage return and line feed.
340A4	\$_RAD	"RAD"
340B4	\$_GRAD	"GRAD"
33FF2	\$_XYZ	"XYZ"
33FE2	\$_R <z< td=""><td>"R∠Z"</td></z<>	"R∠Z"
		"R <angle>Z"</angle>
33FD2	\$_R<<	" <b>R</b> ∠∠"
		"R <angle><angle>"</angle></angle>
34076	\$_EXIT	"EXIT"
34064	\$_ECHO	"ECHO"
34088	<pre>\$_Undefined</pre>	"Undefined"
34002	\$_<<>>	" <b>«»</b> "
34010	\$_{}	"}"
3401E	\$_[]	"[]"
3402C	\$_''	
		Two single quotes.
3403A	\$_::	"::"
34048	\$_LRParens	"()"
34056	\$_2DQ	
		Two double quotes.
33B91	tok,	n n 5
33B85	tok'	"""
		One single quote.
33BFD	tok-	"_"
33B9D	tok.	•
2D848	tok_g	"g"
2D86D	tok_m	"m"
2D8AD	tok_s	"s"
33A77	tok{	"{"
33AD7	tok<<	"«"
33C09	tok=	"="
272D9	tok->	"→"
33C4D	tok0	"0"
33C59	tok1	"1"
33CAD	tok8	"8"
33CB9	tok9	"9"
33ABF	tokESC	"<ESC $>"$
		Escape character.

33B79	Addr.	Name	Description
One double quote.  33A8F toksharp "#"  33AA7 (tok\$) "\$"  33AB3 (tok&) "&"  33BD9 (tok*) "*"  33BF1 (tok+) "+"  33BE5 (tok/) "/"  33C65 (tok2) "2"  33C71 (tok3) "3"  33C7D (tok4) "4"  33C89 (tok5) "5"  33C95 (tok6) "6"  33CA1 (tok7) "7"  33BA9 (tok;) "8"  33ACB (tok>>) "»"  33AEF (tokanglesign) "\( \alpha \)"  33CD (tokCTGROB) "GROB"  33C3F (tokCTSTR) "C\$"  33C1 (tokDER) "\( \alpha \)"  33BB5 (toklparen) "("  33BB5 (toklparen) "("  33BF1 (toksigma) "\( \alpha \)"  33AFB (toksigma) "\( \alpha \)"  33AFB (toksigma) "\( \alpha \)"  33AFB (toksigma) "\( \alpha \)"  33BB5 (toklparen) "("  33BC1 (toksigma) "\( \alpha \)"  33AFB (toksigma) "\( \alpha \)"  33AFB (tokuscore) "\( \alpha \)"  33ABB (tokuscore) "\( \alpha \)"  33ABB (tokylere) "I"  33ABG (tok) "I"	33AE3	tokexponent	"E"
33A8F toksharp "#"  33AA7 (tok\$) "\$"  33AB3 (tok&) "&"  33BD9 (tok*) "*"  33BF1 (tok+) "+"  33BE5 (tok/) "/"  33C65 (tok2) "2"  33C71 (tok3) "3"  33C7D (tok4) "4"  33C89 (tok5) "5"  33C95 (tok6) "6"  33CA1 (tok7) "7"  33BA9 (tok7) "8"  33ACB (tok>) "8"  33ACB (tok>>) "»"  33AEF (tokanglesign) "\( \alpha \)"  33C2D (tokCTGROB) "GROB"  33C3F (tokCTSTR) "C\$"  33B45 (\$DER) "der"  33B45 (\$DER) "der"  33B51 (tokparen) "("  33B61 (toksQRT) "\( \alpha \)"  33B61 (toksQRT) "\( \alpha \)"  33B61 (tokunknown) "UNKNOWN"  33B65 (tok[) "["  33B65 (tok]) "I"  33B66 (tok]) "I"  33B61 (tok) "I"	33B79	tokquote	IIIII
33AA7 (tok\$) "\$"  33AB3 (tok&) "&"  33BD9 (tok*) "*"  33BF1 (tok+) "+"  33BE5 (tok/) "/"  33C65 (tok2) "2"  33C71 (tok3) "3"  33C7D (tok4) "4"  33C89 (tok5) "5"  33C95 (tok6) "6"  33CA1 (tok7) "7"  33BA9 (tok7) "8"  33ACB (tok7) "8"  33ACB (tok7) "8"  33ACB (toko>) "»"  33AEF (tokanglesign) "∠"  33C2D (tokCTGROB) "GROB"  33C3F (tokDER) "∂"  33B45 (\$DER) "der"  33B51 (tokparen) "("  33BC1 (tokparen) "("  33BC1 (tokparen) ")"  33AFB (toksGRT) "∑"  33C15 (toksQRT) "∑"  33B61 (tokuNKNOWN) "UNKNOWN"  33A9B (tokuscore) "."  33B65 (tok) "I"  33B66 (tok]) "I"  33A66 (tok]) "I"  33B61 (tok) "]"  33B61 (tok) "]"  33B61 (tok) "I"			=
33AB3 (tok&) "&" 33BD9 (tok*) "*" 33BF1 (tok+) "+" 33BE5 (tok/) "/" 33C65 (tok2) "2" 33C71 (tok3) "3" 33C7D (tok4) "4" 33C89 (tok5) "5" 33C95 (tok6) "6" 33CA1 (tok7) "7" 33BA9 (tok;) "8" 33ACB (tok>>) "»" 33AEF (tokanglesign) "∠" 33C2D (tokCTGROB) "GROB" 33C3F (tokCTSTR) "C\$" 33C21 (tokDER) "∂" 33B45 (\$DER) "der" 33BB5 (toklparen) "(" 33BF6 (toksorn) "∑" 33AF8 (toksorn) "∑" 33AF8 (toksorn) "∑" 33BF1 (toksorn) "∫" 33BF1 (tokunknown) "UNKNOWN" 33A9B (tokuscore) "_" 33BO7 (tokWHERE) "1" 33A6B (tok] "[" 33A51 (tok]) "[" 33B51 (tok]) "[" 33B51 (tok]) "[" 33B61 (toky)] 33B61 (toky) """	33A8F	toksharp	
33BD9 (tok*) "*"  33BF1 (tok+) "+"  33BE5 (tok/) "/"  33C65 (tok2) "2"  33C71 (tok3) "3"  33C7D (tok4) "4"  33C89 (tok5) "5"  33C95 (tok6) "6"  33CA1 (tok7) "7"  33BA9 (tok;) "8"  33ACB (tok>>) "*"  33AEF (tokanglesign) "\( \alpha \)"  33C2D (tokCTGROB) "GROB"  33C3F (tokCTSTR) "C\$"  33C3F (tokDER) "\( \alpha \)"  33B45 (\$DER) "der"  33B45 (\$DER) "der"  33B61 (tokrparen) "("  33AFB (tokSIGMA) "\( \alpha \)"  33C15 (tokSQRT) "\( \alpha \)"  33B61 (tokUNKNOWN) "UNKNOWN"  33A9B (tokuscore) "-"  33B07 (tokWHERE) "1"  33A6B (tok]) "["  33A51 (tok]) "]"  33BCD (tok^*) "\( \alpha \)"  """	33AA7	(tok\$)	
33BF1 (tok+) "+" 33BE5 (tok/) "/" 33C65 (tok2) "2" 33C71 (tok3) "3" 33C7D (tok4) "4" 33C89 (tok5) "5" 33C95 (tok6) "6" 33CA1 (tok7) "7" 33BA9 (tok;) "8" 33ACB (tok>>) "»" 33AEF (tokanglesign) "∠" 33C2D (tokCTGROB) "GROB" 33C3F (tokDER) "∂" 33B45 (\$DER) "der" 33BB5 (tok]paren) "(" 33BF1 (toksigma) "∑" 33CF1 (toksigma) "J"	33AB3	(tok&)	"&"
33BE5 (tok/) "/" 33C65 (tok2) "2" 33C71 (tok3) "3" 33C7D (tok4) "4" 33C89 (tok5) "5" 33C95 (tok6) "6" 33CA1 (tok7) "7" 33BA9 (tok;) "8" 33ACB (tok>) "»" 33AEF (tokanglesign) "∠" 33C2D (tokCTGROB) "GROB" 33C3F (tokDER) "∂" 33B45 (\$DER) "der" 33B85 (toklparen) "(" 33BF5 (toklparen) "(" 33BF6 (toksigma) "∑" 33AFB (toksigma) "∑" 33AFB (toksigma) "∑" 33B61 (tokunknown) "UNKNOWN" 33A9B (tokuscore) "_" 33B67 (tokwhere) " " 33A6B (tok] "[" 33A51 (tok]) "]" 33BCD (tok^) ">"""	33BD9	(tok*)	"*"
33C65 (tok2) "2"  33C71 (tok3) "3"  33C7D (tok4) "4"  33C89 (tok5) "5"  33C95 (tok6) "6"  33CA1 (tok7) "7"  33BA9 (tok;) "8"  33ACB (tok>>) "»"  33AEF (tokanglesign) "∠"  33C2D (tokCTGROB) "GROB"  33C3F (tokDER) "∂"  33B45 (\$DER) "der"  33B5 (toklparen) "("  33BC1 (tokrparen) ")"  33AFB (toksIGMA) "∑"  33C15 (tokSQRT) "√"  33B61 (tokUNKNOWN) "UNKNOWN"  33A9B (tokuscore) "_"  33B67 (tokWHERE) "1"  33A6B (tok] "["  33A51 (tok]) "]"  33BCD (tok^) "0"	33BF1	(tok+)	
33C71 (tok3) "3" 33C7D (tok4) "4" 33C89 (tok5) "5" 33C95 (tok6) "6" 33CA1 (tok7) "7" 33BA9 (tok;) "8" 33ACB (tok>>) "»" 33AEF (tokanglesign) "∠" 33C2D (tokCTGROB) "GROB" 33C3F (tokCTSTR) "C\$" 33C21 (tokDER) "∂" 33B45 (\$DER) "der" 33B45 (\$DER) "der" 33BC1 (tokrparen) ")" 33AFB (tokSIGMA) "∑" 33C15 (tokSQRT) "√" 33B61 (tokUNKNOWN) "UNKNOWN" 33A9B (tokuscore) "_" 33A6B (tok[) "[" 33A51 (tok]) "]" 33A51 (tok]) "]" 33A51 (tok]) "]" 33BCD (tok^) "^"	33BE5	(tok/)	
33C7D (tok4) "4"  33C89 (tok5) "5"  33C95 (tok6) "6"  33CA1 (tok7) "7"  33BA9 (tok;) "8"  33ACB (tok>>) "»"  33AEF (tokanglesign) "∠"  33C2D (tokCTGROB) "GROB"  33C3F (tokCTSTR) "C\$"  33C21 (tokDER) "∂"  33B45 (\$DER) "der"  33BB5 (toklparen) "("  33BC1 (tokrparen) ")"  33AFB (tokSIGMA) "∑"  33C15 (tokSQRT) "√"  33B61 (tokUNKNOWN) "UNKNOWN"  33A9B (tokuscore) "_"  33A6B (tok[) "["  33A51 (tok]) "]"  33BCD (tok^) "^"	33C65	(tok2)	"2"
33C89 (tok5) "5" 33C95 (tok6) "6" 33CA1 (tok7) "7" 33BA9 (tok;) "8" 33ACB (tok>>) "»" 33AEF (tokanglesign) "∠" 33C2D (tokCTGROB) "GROB" 33C3F (tokCTSTR) "C\$" 33B45 (\$DER) "der" 33BB5 (toklparen) "(" 33BC1 (tokrparen) ")" 33AFB (tokSIGMA) "∑" 33C15 (tokSQRT) "√" 33B61 (tokUNKNOWN) "UNKNOWN" 33A9B (tokuscore) """ 33A6B (tok[) "[" 33A51 (tok]) "]" 33BCD (tok^) "^"	33C71	(tok3)	"3"
33C95 (tok6) "6"  33CA1 (tok7) "7"  33BA9 (tok;) "8"  33ACB (tok>>) "»"  33AEF (tokanglesign) "∠"  33C2D (tokCTGROB) "GROB"  33C3F (tokCTSTR) "C\$"  33C21 (tokDER) "∂"  33B45 (\$DER) "der"  33BB5 (toklparen) "("  33BC1 (tokrparen) ")"  33AFB (tokSIGMA) "Σ"  33C15 (tokSQRT) "√"  33B61 (tokUNKNOWN) "UNKNOWN"  33A9B (tokwHERE) "1"  33A6B (tok] "["  33A51 (tok]) "["  33BCD (tok^) "^"	33C7D	(tok4)	"4"
33CA1 (tok7) "7"  33BA9 (tok;) "8"  33ACB (tok>>) "»"  33AEF (tokanglesign) "∠"  33C2D (tokCTGROB) "GROB"  33C3F (tokCTSTR) "C\$"  33C21 (tokDER) "∂"  33B45 (\$DER) "der"  33BB5 (toklparen) "("  33BC1 (tokrparen) ")"  33AFB (tokSIGMA) "Σ"  33C15 (tokSQRT) "√"  33B61 (tokUNKNOWN) "UNKNOWN"  33A9B (tokuscore) "_"  33A6B (tok[) "["  33A51 (tok]) "]"  33BCD (tok^) "^"	33C89	(tok5)	
33BA9 (tok;) "8"  33ACB (tok>>) "»"  33AEF (tokanglesign) "∠"  33C2D (tokCTGROB) "GROB"  33C3F (tokCTSTR) "C\$"  33B45 (\$DER) "der"  33B85 (toklparen) "("  33BC1 (tokrparen) ")"  33AFB (toksIGMA) "∑"  33C15 (toksQRT) "√"  33B61 (tokunknown) "UNKNOWN"  33A9B (tokuscore) "_"  33A6B (tok[) "["  33A6B (tok]) "]"  33BCD (tok^) "^"	33C95	(tok6)	"6"
33ACB (tok>>) "»"  33AEF (tokanglesign) "∠"  33C2D (tokCTGROB) "GROB"  33C3F (tokCTSTR) "C\$"  33C21 (tokDER) "∂"  33B45 (\$DER) "der"  33B55 (toklparen) "("  33BC1 (tokrparen) ")"  33AFB (tokSIGMA) "Σ"  33C15 (tokSQRT) "√"  33B61 (tokUNKNOWN) "UNKNOWN"  33A9B (tokuscore) "_"  33B07 (tokWHERE) " "  33A6B (tok[) "["  33A51 (tok]) "]"  33BCD (tok^) "^"	33CA1	(tok7)	"7"
33AEF (tokanglesign) "∠"  33C2D (tokCTGROB) "GROB"  33C3F (tokCTSTR) "C\$"  33B45 (\$DER) "der"  33B85 (toklparen) "("  33BC1 (tokrparen) ")"  33AFB (tokSIGMA) "∑"  33C15 (tokSQRT) "√"  33B61 (tokUNKNOWN) "UNKNOWN"  33A9B (tokuscore) "_"  33A6B (tok[) "["  33A6B (tok]) "]"  33BCD (tok^) "^"	33BA9	(tok;)	"8"
33C2D (tokCTGROB) "GROB"  33C3F (tokCTSTR) "C\$"  33C21 (tokDER) "∂"  33B45 (\$DER) "der"  33BB5 (toklparen) "("  33BC1 (tokrparen) ")"  33AFB (tokSIGMA) "Σ"  33C15 (tokSQRT) "√"  33B61 (tokUNKNOWN) "UNKNOWN"  33A9B (tokuscore) "_"  33B07 (tokWHERE) "I"  33A6B (tok] "["  33BCD (tok^) "^"	33ACB	(tok>>)	"»"
33C3F (tokCTSTR) "C\$"  33C21 (tokDER) "∂"  33B45 (\$DER) "der"  33BB5 (toklparen) "("  33BC1 (tokrparen) ")"  33AFB (tokSIGMA) "Σ"  33C15 (tokSQRT) "√"  33B61 (tokUNKNOWN) "UNKNOWN"  33A9B (tokuscore) "_"  33B07 (tokWHERE) "I"  33A6B (tok] "["  33A51 (tok] "]"  33BCD (tok^) "^"	33AEF	(tokanglesign)	" <i>\</i> _"
33C21 (tokDER) "∂" 33B45 (\$DER) "der" 33BB5 (toklparen) "(" 33BC1 (tokrparen) ")" 33AFB (tokSIGMA) "Σ" 33C15 (tokSQRT) "√" 33B61 (tokUNKNOWN) "UNKNOWN" 33A9B (tokuscore) "_" 33B07 (tokWHERE) " " 33A6B (tok[) "[" 33A51 (tok]) "]" 33BCD (tok^) "^"	33C2D	(tokCTGROB)	"GROB"
33B45 (\$DER) "der"  33BB5 (toklparen) "("  33BC1 (tokrparen) ")"  33AFB (tokSIGMA) "Σ"  33C15 (tokSQRT) "√"  33B61 (tokUNKNOWN) "UNKNOWN"  33A9B (tokuscore) "_"  33B07 (tokWHERE) " "  33A6B (tok[) "["  33A51 (tok]) "]"  33BCD (tok^) "^"	33C3F	(tokCTSTR)	"C\$"
33BB5 (toklparen) "(" 33BC1 (tokrparen) ")" 33AFB (tokSIGMA) "Σ" 33C15 (tokSQRT) "√" 33B61 (tokUNKNOWN) "UNKNOWN" 33A9B (tokuscore) "_" 33B07 (tokWHERE) " " 33A6B (tok[) "[" 33A51 (tok]) "]" 33BCD (tok^) "^"	33C21	(tokDER)	"∂"
33BC1 (tokrparen) ")" 33AFB (tokSIGMA) "Σ" 33C15 (tokSQRT) "√" 33B61 (tokUNKNOWN) "UNKNOWN" 33A9B (tokuscore) "_" 33B07 (tokWHERE) "I" 33A6B (tok[) "[" 33A51 (tok]) "]" 33BCD (tok^) "^"	33B45	(\$DER)	"der"
33AFB (tokSIGMA) "Σ" 33C15 (tokSQRT) "√" 33B61 (tokUNKNOWN) "UNKNOWN" 33A9B (tokuscore) "_" 33B07 (tokWHERE) " " 33A6B (tok[) "[" 33A51 (tok]) "]" 33BCD (tok^) "^"	33BB5	(toklparen)	"("
33C15 (tokSQRT) "\" 33B61 (tokUNKNOWN) "UNKNOWN"  33A9B (tokuscore) "_"  33B07 (tokWHERE) "I"  33A6B (tok[) "["  33A51 (tok]) "]"  33BCD (tok^) "^"	33BC1	(tokrparen)	")"
33B61 (tokUNKNOWN) "UNKNOWN"  33A9B (tokuscore) "_"  33B07 (tokWHERE) "I"  33A6B (tok[) "["  33A51 (tok]) "]"  33BCD (tok^) "^"	33AFB	(tokSIGMA)	" <u>\Sigma"</u>
33A9B (tokuscore) "_" 33B07 (tokWHERE) " " 33A6B (tok[) "[" 33A51 (tok]) "]" 33BCD (tok^) "^"	33C15	(tokSQRT)	",/"
33A9B (tokuscore)  33B07 (tokWHERE) "I"  33A6B (tok[) "["  33A51 (tok]) "]"  33BCD (tok^) "^"	33B61	(tokUNKNOWN)	"UNKNOWN"
33A6B (tok[) "["  33A51 (tok]) "]"  33BCD (tok^) "^"	33A9B	(tokuscore)	" " —
33A51 (tok]) "]" 33BCD (tok^) "^"	33B07	(tokWHERE)	" "
33BCD (tok^) "^"	33A6B	(tok[)	"["
33BCD (COK)	33A51	(tok])	"]"
	33BCD	(tok^)	IIAII
33A83 (tok}) "}"	33A83	(tok})	"}"
33B13 (14SPACES\$) "" String of 14 spaces.	33B13	•	

# 6.1.3 Built-in Strings with Stack Manipulation

Addr.	Name	Description
35D94	NULL\$SWAP	$( ob \rightarrow $ ob )$
		NULL\$, then SWAP.
04D3E	DROPNULL\$	( ob $ ightarrow$ NULL\$ )
		DROP then NULL\$.
25EEC	NULL\$TEMP	$( \rightarrow \$ )$
		Creates null string in temporary memory (NULL\$,
		then TOTEMPOB).

#### 6.1.4 Conversion

Addr.	Name	Description
25F77	#>\$	( ♯ → \$ )
		Creates string from the bint (decimal).
25F72	#:>\$	( # -> "#: <u>"</u> " )
		Creates string from the bint and appends a colon and a
		space. Ex: "1:_"
25F0F	a%>\$	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
		Converts real number into string using current display
		mode.
		aka: a%>\$,
05BE9	ID>\$	( $id/lam \rightarrow $$ )
		Converts identifier into string.
25EB3	DOCHR	( % → \$ )
		Creates string of the character with the number speci-
		fied.
0F1006	^Z>S	$(Z \rightarrow S)$
		Converts Z into a string (decimal).
2EFC1	hxs>\$	( hxs $\rightarrow$ \$ )
		Uses current display mode and wordsize.
2EFC0	HXS>\$	( hxs $\rightarrow$ \$ )
		Does hxs>\$ and then appends base character.

# 6.1.5 Management

Addr.	Name	Description
05A75	#>CHR	$( \# \rightarrow chr )$
		Returns character with the specified ASCII
		code.
37AA5	CHR>\$	( chr $ ightarrow$ \$* Strings )
		Converts a character into a string.
05636	LEN\$	( $\$ \rightarrow \$ length$ )
		Returns length in bytes.
357E2	DUPLEN\$	( \$ → \$ # )
		DUP then LEN\$.
05622	OVERLEN\$	( $\$$ ob $ o$ $\$$ ob $\#len$ )
		OVER then LEN\$.
361DA	NEWLINE\$&\$	( $$ \rightarrow "$\0a" )$
		Appends newline character to string.
		aka: NEWLINE&\$
2F31A	APNDCRLF	( \$ → \$' )
		Appends carriage return and line feed to
		string.
050ED	CAR\$	( $\$ \rightarrow \text{chr}$ )
		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.
378FA	POS\$	( $\$$ $\$$ find start $\# \to \#$ pos )
		Search for \$find in \$search, starting at posi-
		tion #start. Returns position of \$find or 0 if not
		found. Same entry as POSCHR.
378FA	POSCHR	( $\$$ search chr $\#$ start $\to$ $\#$ pos )
		Same entry as POS\$.
37906	POS\$REV	( $\$$ $\$$ find $\#$ limit $\to$ $\#$ pos )
		Searches backwards from #limit to #1. Same
		entry as POSCHRREV.
37906	POSCHRREV	( $seach\ chr\ \#start\  o\ \#pos\ )$
		Same entry as POS\$REV.

25EA0 COERCE\$22 (\$ → \$')  If the string is longer than 22 characters, truncates it to 21 characters and appends "".  2F16D Blank\$ (#len → \$)  Creates a string with the specified number of spaces.  2EEF0 PromptIdUtil (id ob → \$)  Creates string of the form "id: ob".  25EF8 SEP\$NL (\$ → \$' \$'')  Separates string at the first newline. \$" is the substring before the first newline; \$' the substring after the first newline.
cates it to 21 characters and appends "".   2F16D Blank\$ ( $\#len \rightarrow \$$ ) Creates a string with the specified number of spaces.   2EEF0 PromptIdUtil ( $id ob \rightarrow \$$ ) Creates string of the form "id: ob".   25EF8 SEP\$NL ( $\$ \rightarrow \$' \$''$ ) Separates string at the first newline. $\$''$ is the substring before the first newline; $\$'$ the substring after the first newline.
2F16D Blank\$ ( #len → \$ ) Creates a string with the specified number of spaces.  2EEF0 PromptIdUtil (id ob → \$ ) Creates string of the form "id: ob".  25EF8 SEP\$NL (\$ → \$ ' \$ ' ' ) Separates string at the first newline. \$" is the substring before the first newline; \$' the substring after the first newline.
Creates a string with the specified number of spaces. 2EEF0 PromptIdUtil (id ob $\rightarrow$ \$) Creates string of the form "id: ob". 25EF8 SEP\$NL (\$ $\rightarrow$ \$' \$'') Separates string at the first newline. \$" is the substring before the first newline; \$' the substring after the first newline.
spaces.   2EEF0 PromptIdUtil (id ob $\rightarrow$ \$) Creates string of the form "id: ob".  25EF8 SEP\$NL (\$ $\rightarrow$ \$' \$'') Separates string at the first newline. \$" is the substring before the first newline; \$' the substring after the first newline.
2EEF0 PromptIdUtil (id ob → \$)  Creates string of the form "id: ob".  25EF8 SEP\$NL (\$→\$'\$'')  Separates string at the first newline. \$" is the substring before the first newline; \$' the substring after the first newline.
Creates string of the form "id: ob".  25EF8 SEP\$NL (\$ \( \display \) \$' \( \display \) ' \( \display \) Separates string at the first newline. \$" is the substring before the first newline; \$' the substring after the first newline.
25EF8 SEP\$NL ( $\$ \rightarrow \$' \$''$ ) Separates string at the first newline. $\$''$ is the substring before the first newline; $\$'$ the substring after the first newline.
Separates string at the first newline. \$" is the substring before the first newline; \$' the substring after the first newline.
substring before the first newline; \$' the substring after the first newline.
string after the first newline.
<u>g</u>
09A003 (^WRAP\$) ( $\$$ #width $\rightarrow$ $\$$ ' )
Replace SPACE chars with NEWLINE in or-
der to fit the text in the given #width. Used by
ViewStrObject. Very fast (bang type).
05733 SUB\$ ( \$ $\#$ start $\#$ end $\to$ \$' )
Returns substring between specified positions.
3628E #1-SUB\$ ( \$ $\#$ start $\#$ end+ $\#$ 1 $\to$ \$' )
Does #1- and then SUB\$.
362A2 1_#1-SUB\$ ( \$ $\#$ end $\rightarrow$ \$' )
Returns substring from the first character to
the character before the specified position.
aka: 1_#1-SUB
362B6 LAST\$ ( \$ $\#$ start $\rightarrow$ \$' )
Returns substring from the specified start po-
sition to the end (inclusive).
362CA #1+LAST\$ ( \$ $\#$ start- $\#$ 1 $\to$ \$' )
Returns substring from the specified start po-
sition to the end (exclusive).
35DA8 SUB\$SWAP (ob \$ $\# \#' \rightarrow \$'$ ob )
SUB\$ then SWAP.
2A5CA SUB\$1 $\sharp$ (\$ $\sharp$ $\rightarrow$ $\sharp$ ')
Returns bint with ASCII code of character at
the specified position.

Addr.	Name	Description
34C82	EXPAND	( hxs #nibs -> hxs' )
		Appends null characters to the string. Since
		refers to the number of nibbles, you must use
		a number twice as large as the number of null
		characters you want appended.
05193	&\$	<pre>( \$ \$' → \$+\$' )</pre>
		Concatenates two strings.
36FF6	&\$SWAP	( ob $\$ \$' \rightarrow \$+\$'$ ob )
		&\$ then SWAP.
353CD	!append\$	$(\ \ \$\ \ \ \ \ \ \ \ \ \ \ \ )$
		Tries &\$, if not enough memory does !!ap-
		pend\$?.
3533C	!insert\$	$(\ \ \$\ \ \ \ \ \ )$
		Does SWAP then !append\$.
35F6A	!append\$SWAP	( ob $\$ \$' \rightarrow \$+\$'$ ob )
		!append\$ then SWAP.
35369	!!append\$?	( \$ \$' → \$+\$' )
		Attempts append "in place" if target is in tem-
		pob.
353F7	!!append\$	$(\ \ \$\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $
		Tries appending "in place".
353EB	!!insert\$	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $
0505-	+	Tries inserting "in place".
0525B	>H\$	( \$ chr → \$' )
0.5.055	\ m.d	Prepends character to string
052EE	>T\$	( \$ chr → \$' )
25555		Appends character to string.
35BD7	APPEND_SPACE	$( \ \$ \rightarrow \$' \ )$
25246	CIVIA D C d	Appends space to string.
35346	SWAP&\$	$(\$ \$' \rightarrow \$' + \$)$ Concatenates two strings.
כתשיםט	<b>ТТМЕСТО</b>	( %dt %tm → "dy dt tm" )
2EED3	TIMESTR	Returns string representation of time, using
		current format. Example:
		"WED 06/24/98 10:00:45A"
25E7C	AND\$	(\$1 \$2 → \$')
١٠ تار ت	иνυγ	Logical AND. Errors if strings are not the same
		length.
		iengui.

Addr.	Name	Description
25EF0	OR\$	( \$ \$' → \$'' )
		Logical OR. Errors if strings are not the same
		length.
25F0D	XOR\$	( \$ \$' \rightarrow \$'' )
		Logical XOR. Errors if strings are not the same
		length.
2F1A7	CHARSEDIT	$( \rightarrow )$
		HP49 character browser. This is an interac-
		tive application from which characters can be
		echoed into the command line.

# 6.1.6 Parsing Strings

Addr.	Name	Description
25EB7	DOSTR>	( \$ → ? )
		Internal version of STR $\rightarrow$ .
2EF62	palparse	$( \ \$ \rightarrow \ ob \ \mathtt{T} \ )$
		( $\$ \rightarrow \$$ #pos $\$$ ' F )
		Tries parsing a string into an object. If success-
		ful, returns object and TRUE, otherwise returns
		position of error, the offending part of the string
		\$', and FALSE. If the string contains several ar-
		guments, the resulting object is a secondary con-
		taining these objects.
25E68	!*trior	$( F \rightarrow \langle SKIP \rangle )$
		$(TT \rightarrow \langle COLA \rangle)$
25E67	!*triand	$(TT \rightarrow)$
		$( FT \rightarrow FT < SEMI > )$
26206	tok8cktrior	$(\$1 \$1 \rightarrow :: \$1 < 0b1 > ;)$
		$(\$1 \$2 \rightarrow :: \$1 < 0b2 > < Rest > ;)$
261BB	tok8trior	( GNT data \$1 \$1 $ ightarrow$ :: GNT data Get-
		NextToken ; )
		( GNT data \$1 \$2 $\rightarrow$ :: \$1 <0b1>
00-65	<b>7</b>	<rest> ; )</rest>
29E67	nultrior	( NULL\$ → :: ; )
0.5		$(\$ \rightarrow :: \$ < 0b1 > < Rest > ;)$
25EDB	GetNextToken	( =====
		#next \$token )

Addr.	Name	Description
2F33C	getmatchtok	( hxs-mask \$ $\sharp$ loc \$_tok $\to$ hxs-mask \$
		<pre>#next \$match )</pre>
2EF6E	ParseFail	( ob \$parsed #pos \$' $ ightarrow$ )
		Uses DispBadToken to re-edit the parsed string
		and displays "Syntax Error".
2EF6F	DispBadToken	( ob \$parsed #pos \$' $ ightarrow$ )
		Re-edits the parsed string, positions the cursor
		to the location of the error. Used by ParseFail.

# 6.1.7 Decompilation

Addr.	Name	Description
2F191	!DcompWidth	( ♯ → )
		Sets the width (in characters) of decompiled
		strings. This width is used to cut the re-
		sulting 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 stkde-
		comp\$w and editdecomp\$w to make sure
0-100	B 77 11 1 0	the current width is used and not changed.
2F190	DcompWidth@	$(\rightarrow \#)$
		Recalls the width of decompiled strings (in
		characters).
26459	setStdWid	$(\  ightarrow\ )$
		Sets DcompWidth to the standard value for
		stack display, either 19 or 30 characters, de-
		pending on system flag 72 (stack minifont).
2645E	setStdEditWid	$( \rightarrow )$
		Sets DcompWidth to the width for editing,
		either 21 or 32 characters, depending on sys-
		tem flag 73 (edit minifont).
25F13	stkdecomp\$w	$( ob \rightarrow \$ )$
		Decompiles for stack display using the cur-
		rent DcompWidth to cut the string if it is too
		long.
		- 0.

Addr.	Name	Description
25E6D	1stkdecomp\$w	( ob $\rightarrow$ \$ )
		Calls setStdWid and decompiles for stack
		display (cutting the string if necessary).
2A842	Decomp1Line	$( ob \rightarrow \$ )$
		Same as 1stkdecomp\$w.
2A904	RPNDecomp1Line	$( ob \rightarrow \$ )$
		Same as DecomplLine but enforce RPN
		mode (system flag 95 clear) during execu-
05565	` D ' ' ' '	tion.
25E6F	>Review\$	( $id \rightarrow \$$ )
		Makes a string from the variable name
		and its contents (decompiled with De-
		complLine), for display with the review key.
2A8E4	DecompStd1Line32	( ob $\rightarrow$ \$ )
ZAOLI	ресопраситите за	Sets 32 as DcompWidth and decompiles us-
		ing stkdecomp\$w.
2A9C4	RPNDecompStd1Line32	$( ob \rightarrow \$ )$
		Same as DecompStd1Line32 but enforce
		RPN mode (system flag 95 clear) during ex-
		ecution.
2A8C9	DecompStd1Line	( ob $ ightarrow$ \$ )
		Calls setStdWid and decompiles, cutting if
		the string becomes too long.
2A9A4	RPNDecompStd1Line	( ob $\rightarrow$ \$ )
		Same as DecompStd1Line but enforce RPN
		mode (system flag 95 clear) during execu-
		tion.
2A893	Decomp#Disp	$( ob \# \rightarrow \$ )$
		Calls setStdWid and decompiles ob (User-
		RPL components only), breaks the string
		into lines and returns the first #+1 lines.
27064	DDND	Used for multiline display in stack level 1. ( ob $\# \rightarrow \$$ )
2A964	RPNDecomp#Disp	Same as Decomp#Disp but enforce RPN
		mode (system flag 95 clear) during execu-
		tion.
		VIVII.

Addr.	Name	Description
2A878	Decomp#Line	( ob # → \$ ) Similar to Decomp#Disp, but the returned string is an internal representation of the different lines to be displayed. Used for mul- tiline display in stack level 1.
2A944	RPNDecomp#Line	( ob # $\rightarrow$ \$ ) Same as Decomp#Line but enforce RPN mode (system flag 95 clear) during execution.
25F11	editdecomp\$w	( ob → \$ ) Decompiles entire object for editing. It only decompiles the UserRPL components. Some System RPL entries like TakeOver are simply skipped, others are written as "External". Breaks the resulting strings into lines using the current DcompWidth.
25ECE	EDITDECOMP\$	( ob $\rightarrow$ \$ ) Calls setStdEditWid and the decompiles for editing like editdecomp\$w.
2A85D	DecompEdit	( ob $ ightarrow$ $\$$ ) Same as <code>EDITDECOMP</code> $\$$ .
2A924	RPNDecompEdit	( ob $\rightarrow$ \$ ) Same as DecompEdit but enforce RPN mode (system flag 95 clear) during execution.
2AA43	AlgDecomp	( ob $\rightarrow$ \$ ) Calls DecompEdit with a few checks around it.
25EAA	DECOMP\$	( ob $\rightarrow$ \$ ) Calls setStdWid and decompiles entire object (UserRPL components only). Breaks the string into lines using DcompWidth as width.
39CB3	(ob&\$)	( ob \$ $\rightarrow$ "ob\$" ) Applies DECOMP\$ to ob and concatenates with the string.
39C9F	(\$&ob)	( $\$$ ob $\to$ " $\$$ ob" ) Applies DECOMP $\$$ to ob and concatenates with the string.

Addr.	Name	Description
25EB1	DO>STR	(\$ → \$)
		$( ob \rightarrow $ )$
		Internal version of $\rightarrow$ STR.
1A7006	^DO>STRID	$(id/ob \rightarrow $)$
	_	Like DO>STR but without quotes for id.
2A8AE	DecompEcho	$( ob \rightarrow \$ )$
		Calls setStdEditWid and decompiles the
		entire object (UserRPL only) into a single
27004	DDNDogomoEgho	line. ( ob $ ightarrow$ \$ )
2A984	RPNDecompEcho	Same as DecompEcho but enforce RPN
		mode (system flag 95 clear) during execu-
		tion.
2F1BF	Decomp%Short	( % #width $\rightarrow$ \$ )
21 121	Decomp volice	Decompiles a real number into a string of
		the given #width. It will drop less signifi-
		cant digits or add zeros as needed, but will
		also exceed #width when necessary. E.g. "-
		1.e-33" cannot be written with less than 7
		characters, so even if #width is less, 7 chars
		will be used. %0 is always decompiled as "0".
001004	^FSTR1	( ob $ ightarrow$ \$ )
		The decompiler used by stkdecomp\$w,
		1stkdecomp\$w, Decomp1Line, Decomp-
		Std1Line32. DcompWidth must be set be-
		fore this is called.
003004	^FSTR3	$( ob \# \rightarrow \$ )$
		The decompiler used by Decomp#Line.
		DcompWidth must be set before this is
004004	^ FI CIED 4	called.
004004	^FSTR4	( ob $\rightarrow$ \$ ) The decompiler used by editdecomp\$w,
		DecompEdit, EDITDECOMP\$. DcompWidth
		must be set before this is called.
005004	^FSTR5	( ob $\rightarrow$ \$ )
00001	151115	The decompiler used by DecompEcho.
		DcompWidth must be set before this is
		called.

Addr.	Name	Description
006004	^FSTR6	( ob $\#$ $\rightarrow$ $\$$ )
		The decompiler used by Decomp#Line.
		DcompWidth must be set before this is
		called.
007004	^FSTR7	( ob $ ightarrow$ \$ )
		The decompiler used by DO>STR. Dcomp-
		Width must be set before this is called.
009004	^FSTR9	( ob $ ightarrow$ \$ )
		The decompiler used by DecompStd1Line.
		DcompWidth must be set before this is
		called.
00D004	^FSTR13	( ob $ ightarrow$ \$ )
		The decompiler used by DECOMP\$. Dcomp-
		Width must be set before this is called.
35B82	palrompdcmp	( romptr $\rightarrow$ \$ T )
		Decompiles a rompointer for the UserRPL
		stack. If it is a named rompointer, returns
		the name. Otherwise returns "XLIB n m".

# 6.1.8 String Tests

Addr.	Name	Description
0556F	NULL\$?	( ob $ ightarrow$ flag )
36252	DUPNULL\$?	( ob $ ightarrow$ ob flag )
2F321	CkChr00	( $\$ \rightarrow \$$ flag )
		Returns FALSE if string contains any null charac-
		ters.

# Chapter 7 Hex Strings

Hexadecimal strings are the numbers that are called Binary Integers in the manual, which can be represented in several bases. In System RPL they are called Hexadecimal Strings. They are created using the structure HXS <len> <hexbody>. len is the length of the string (number of nibbles or hexadecimal digits), in hexadecimal form, and hexbody is the actual contents of it. The tricky part about it is that because of the HP internal architecture, you must enter the contents in reverse order. To get, for example, the hex string #12AD7h, you must enter HXS 5 7DA21. To get #12345678h use HXS 8 87654321. In System RPL, hexadecimal strings can be of any length, unlike in User RPL, where they are limited to 16 nibbles or 64 bits.

To convert an hex string to and from a bint, use the commands  $\tt HXS>\#$  and #>HXS. To convert an  $\tt HXS$  to and from a real number, use #># (or  $\tt HXS>\#$ ) and \$>#.

See below for more commands related to hex strings.

#### 7.1 Reference

#### 7.1.1 Conversion

Addr.	Name	Description
059CC	#>HXS	$( \ \# \rightarrow \ \text{hxs} \ )$
		Length will be five.
2EFCB	%>#	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
		Converts real number into hxs. Should be called %>HXS.

#### 7.1.2 General Functions

Addr.	Name	Description
2EFBE	WORDSIZE	( → # )
		Returns the current wordsize as a bint.
2EFAA	dostws	$( \ \# \  o \ )$
		Sets the current wordsize.
055D5	NULLHXS	HXS 0
		Puts a null hxs in the stack.
0518A	&HXS	( hxs hxs' $\rightarrow$ hxs'' )
		Appends hxs" to hxs'.
34C82	EXPAND	( hxs $\#$ nibs $\rightarrow$ hxs' )
		Appends #nibs zero nibbles to the hxs.
05616	LENHXS	( hxs $\rightarrow$ #nibs )
		Returns length in nibbles.
05815	SUBHXS	( hxs #m #n $\rightarrow$ hxs' )
		Returns sub hxs string.
2EFB9	bit+	( hxs hxs' $\rightarrow$ hxs'' )
		Adds two hxs.
2EFC8	bit%#+	( % $hxs \rightarrow hxs'$ )
		Adds real to hxs, returns hxs.
2EFC9	bit#%+	( hxs % $\rightarrow$ hxs' )
		Adds real to hxs, returns hxs.
2EFBA	bit-	( hxs hxs' $\rightarrow$ hxs'' )
		Subtracts hxs2 from hxs1.
2EFC6	bit%#-	( % $hxs \rightarrow hxs'$ )
		Subtracts hxs from real, returns hxs.
2EFC7	bit#%-	( hxs % $\rightarrow$ hxs' )
		Subtracts real from hxs, returns hxs.
2EFBC	bit*	( hxs hxs' $\rightarrow$ hxs'' )
		Multiplies two hxs.
2EFC4	bit%#*	( % $hxs \rightarrow hxs'$ )
		Multiplies real by hxs, returns hxs.
2EFC5	bit#%*	$( hxs % \rightarrow hxs' )$
		Multiplies hxs by real, returns hxs.
2EFBD	bit/	$( hxs hxs' \rightarrow hxs'' )$
		Divides hxs1 by hxs2.
2EFC2	bit%#/	( % $hxs \rightarrow hxs'$ )
		Divides real by hxs, returns hxs.

58 7. Hex Strings

Addr.	Name	Description
2EFC3	bit#%/	( hxs % $\rightarrow$ hxs' )
		Divides hxs by real, returns hxs.
2EFAC	bitAND	( hxs hxs' $\rightarrow$ hxs'' )
		Bitwise AND.
2EFAD	bitOR	( hxs hxs' $\rightarrow$ hxs'' )
		Bitwise OR.
2EFAE	bitXOR	( hxs hxs' $\rightarrow$ hxs'' )
		Bitwise XOR.
2EFAF	bitNOT	( hxs $\rightarrow$ hxs' )
		Bitwise NOT.
2EFB8	bitASR	( hxs $\rightarrow$ hxs' )
		Arithmetic shift one bit to the right. The most signif-
		icant bit (the sign) does not change.
2EFB6	bitRL	( hxs $\rightarrow$ hxs' )
		Shifts circularly one bit to the left.
2EFB7	bitRLB	$( hxs \rightarrow hxs' )$
		Shifts circularly one byte to the left
2EFB4	bitRR	$( hxs \rightarrow hxs' )$
		Shifts circularly one bit to the right.
2EFB5	bitRRB	$( hxs \rightarrow hxs' )$
		Shifts circularly one byte to the right.
2EFB0	bitSL	$( hxs \rightarrow hxs' )$
		Shifts one bit to the left.
2EFB1	bitSLB	$( hxs \rightarrow hxs' )$
		Shifts one byte to the left.
2EFB2	bitSR	( hxs $\rightarrow$ hxs' )
		Shifts one bit to the right.
2EFB3	bitSRB	( hxs $\rightarrow$ hxs' )
		Shifts one byte to the right.

### 7.1.3 Tests

Addr.	Name	Description
2EFCC	HXS==HXS	( hxs hxs' $ ightarrow$ %flag )
		== test
2F0EE	HXS#HXS	( hxs hxs' $ ightarrow$ %flag )
		$ eq  ext{test}$

Addr.	Name	Description
2EFCF	HXS <hxs< td=""><td>( hxs hxs' → %flag )</td></hxs<>	( hxs hxs' → %flag )
		< test
2EFCD	HXS>HXS	( hxs hxs' $\rightarrow$ %flag )
		> test
2EFCE	HXS>=HXS	( hxs hxs' $\rightarrow$ %flag )
		$\geq  ext{test}$
2F0EF	HXS<=HXS	( hxs hxs' $\rightarrow$ %flag )
		$\leq  ext{test}$

# Chapter 8 Identifiers

Identifiers are used to represent the names of objects stored in memory (i.e., variables). To the user, they appear in the stack between single quotes, that is, ''. In System RPL, they are created with ID <name>. When you use this structure, you do not always get the identifier in the stack. It is always evaluated. So, if variable anumber contains 123.45 and you put somewhere in your program ID anumber, the identifier is evaluated, recalling the contents of the variable. This way, the stack will contain 123.45. To put an id to the stack, use 'ID <name>. As you will see on Chapter 19, the command 'puts the object after it in the stack. This is called *quoting*. However, ID <name> (without the ') will also put the id in the stack if there is no variable called <name>. This is similar to be behaviour you get when you enter the name of a variable without the quotes in the HP49 command line.

You can convert a string to an id using \$>ID. The opposite transformation is archived with ID>\$.

There is also another kind of identifiers: the temporary identifiers, or lams. These are used when creating local variables, and you will learn about them later in Chapter 18. They are created with LAM <name>, and work pretty much like normal ids.

Since ids are closely related to memory access, the functions dealing with its are listed in Chapter 24.

# **Chapter 9 Tagged Objects**

In order to insert a tagged object in your program, use the structure TAG <tag> <object>. Tag is a string without quotes, and the object can be anything. To create 0: 1790, for example, you would use TAG 0 % 1790. An object can have multiple tags, but there is not much use for that.

The word >TAG creates a tagged object, given the object (in level two) and a string representing the tag (in level one). %>TAG works the same way, but tags an object with a real number. ID>TAG tags an object with an identifier. To remove all tags from an object, call STRIPTAGS.

A few more commands related to tagged objects are listed on below.

Note that the programmer seldom needs to worry about tagged objects, because the type dispatching mechanism (which is described in section 29.2) can automatically strip tags from the arguments to your program.

#### 9.1 Reference

Addr.	Name	Description
05E81	>TAG	( ob $\$$ tag $\rightarrow$ tagged )
		Tags an object.
2F266	USER\$>TAG	( ob $tag \rightarrow tagged$ )
		Maximum of 255 characters in string.
2F223	%>TAG	( ob % $ ightarrow$ tagged )
		Converts real to string using current display
		mode and tags object.
05F2E	ID>TAG	( ob id/lam $ ightarrow$ tagged )
		Tags object with identifier or lam.
37B04	TAGOBS	( ob $\$$ tag $ o$ tagged )
		( ob $\{ \ \$ \ \}  o  tagged$ )
		Tags one or more objects.

Addr.	Name	Description
37ABE	STRIPTAGS	( tagged $\rightarrow$ ob )
		Strips all tags from the object.
37AEB	STRIPTAGS12	( tagged ob' $ ightarrow$ ob ob' )
		Strips all tags from the object in level two.

# Chapter 10 Arrays

There are actually two groups of objects that represent arrays in the HP49G. The first group (which will be described in this chapter) has existed since the HP48: the normal arrays (to the user they can be only of real or complex numbers), and the linked arrays, which are not accessible to the user. The HP49 introduced a new kind of object to represent arrays: the Symbolic Matrices. Since these are actually a part of the HP49 CAS, they are described in Chapter 43.

In User RPL, arrays can be only of real or complex numbers. In System RPL, you can have arrays of anything, even arrays of arrays. Note that an array is not a composite object (see Chapter 11), even if it looks like one. Also, an array can only contain one kind of object.

Using MASD, arrays are entered like this:

```
1 ARRY [[ % 1. % 2. %3. ]
[ % 4. % 5. %6. ]]
```

This is not much different from entering an array in the normal HP49 command line.

You can also create an array of (normal, not extended) real or complex numbers by putting them in order in the stack, and entering a list representing the dimensions of the array (real numbers, not bints) in level one. Then run ^XEQ>ARRY. This function does error checks to ensure there are enough arguments and if they are of the supported types.

The function ^ARSIZE returns the number of elements in an array. You can get the dimensions of the array with ^DIMLIMITS, which returns a list of bints representing the array dimensions. To get one element of an array, put the element number in level two, the array in level one, and run GETATELN. You will get the element and TRUE if it was found or only FALSE if the element does not exist. More array functions are listed below.

There is also another kind of array: the linked arrays. Linked arrays are like normal arrays, except that they have a table with pointers to all the

64 10. Arrays

objects in the array. This makes access to array elements faster, because when you need to access one object in the linked array, the only thing necessary is to read the pointer to that object in the table, and go directly there. With normal arrays, a sequential search is necessary.

The entries here all deal with the normal arrays (even though some of them also work for CAS' Symbolic matrices). For entries specific to Symbolic matrices, see Chapter 43.

#### 10.1 Reference

#### 10.1.1 General Functions

Addr.	Name	Description
0371D	GETATELN	( $\#$ [] $\rightarrow$ ob T)
		$( \# [] \rightarrow F)$
		Gets one element from array.
16D006	^MDIMS	( [[]] $ ightarrow$ #rows #cols T )
		( [] $ ightarrow$ #elem F )
		Returns the size of an array. Equivalent to the
		HP48 command MDIMS.
35FD8	MDIMSDROP	( [2D] $\rightarrow$ #m #n )
		MDIMS followed by DROP.
16E006	^DIMLIMITS	$( [] \rightarrow \{ \# \} )$
		$( \hspace{.15cm} [\hspace{.15cm} [\hspace{.15cm} ]\hspace{.15cm}] \hspace{.15cm} \rightarrow \hspace{.15cm} \{ \# \hspace{.15cm} \# \} \hspace{.15cm} )$
		Returns the size of an array, like the User com-
		mand SIZE, but the lengths are bints and not re-
		als. Equivalent to the HP48 command DIMLIM-
		ITS.
35E006	^ARSIZE	$( [] \rightarrow \# )$
		Returns max # in an array.
36183	OVERARSIZE	( [] ob $ ightarrow$ [] ob #elts )
		Does OVER then ARSIZE.
260F8	PULLREALEL	$( [\%] \# \rightarrow [\%] \% )$
		Gets real element.
260F3	PULLCMPEL	$( [C%] \# \rightarrow [C%] C% )$
		Gets complex element.

Addr.	Name	Description
26102	PUTEL	( [%] % # → [%]')
		( [C%] C% $\# \rightarrow [C\%]'$ )
		Puts element at specified position. Converts to
		"short" before. Warning: no copy to tempob first.
26107	PUTREALEL	( [%] % # → [%]')
		Puts real element at specified position. Warning:
		no copy to tempob first.
260FD	PUTCMPEL	( [C%] C% $\# \rightarrow [C\%]'$ )
		Puts complex element at specified position.
		Warning: no copy to tempob first.
33B006	^MATTRAN	( $M \rightarrow M'$ )
		Matrix transposition.
331006	^Yext	( V2 V1 $ ightarrow$ ob )
		Scalar product of symbolic vectors, no check.

#### 10.1.2 Conversion

Addr.	Name	Description
169006	^BESTMATRIXTYPE	( ob $\rightarrow$ ob )
		Converts symbolic matrix with real/cmplex
		entries to a numeric array.
172006	^CKNUMARRY	$( ob \rightarrow ob )$
		Tests if ob is a numeric array. Tries to convert
		symbolic array to numeric array.
178006	^MATRIX2ARRAY	$( \hspace{.15cm} [\hspace{.15cm}] \hspace{.15cm} \rightarrow \hspace{.15cm} [\hspace{.15cm}] \hspace{.15cm} )$
		$( \hspace{.1cm} [\hspace{.1cm} [\hspace{.1cm}]\hspace{.1cm}] \hspace{.1cm} \rightarrow \hspace{.1cm} [\hspace{.1cm} [\hspace{.1cm}]\hspace{.1cm}] \hspace{.1cm})$
		Tries to convert a symbolic matrix to a nu-
		meric one.
001007	^ListToArry	( $\{\}/\{\{\}\}$ $\rightarrow$ []/[[]] TRUE )
		( $\{\}/\{\{\}\}$ $ ightarrow$ FALSE )
		If possible, converts list of lists to normal
		array and returns TRUE. Otherwise, returns
		FALSE.
17F006	^XEQ>ARRY	$(ob1obn {%n} \rightarrow [])$
		( obl1obmn $\{%m %n\} \rightarrow [[mxn]]$ )
		Builds a matrix a la $\rightarrow$ ARRY.
17C006	^XEQARRY>	( [] $ ightarrow$ oblobn meta-arry )
		Explodes a matrix a la $\rightarrow$ ARRY.

66 10. Arrays

Addr.	Name	Description
002007	^ArryToMatrix	$( [] \rightarrow M )$
		Converts array to symbolic array.

#### 10.1.3 Statistics

Addr.	Name	Description
2EEDA	STATCLST	$(\  ightarrow\ )$
		Clears $\Sigma$ DAT.
2EEDB	STATSADD%	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
		Internal $\Sigma$ +.
2EEDC	STATN	$( \rightarrow N )$
		Internal N $\Sigma$ .
2EEDF	STATSMIN	$( \rightarrow \% )$
		Internal MIN $\Sigma$ .
2EEDD	STATSMAX	$( \rightarrow \% )$
		Internal MAX $\Sigma$ .
2EEDE	STATMEAN	$( \rightarrow \% )$
		$( \rightarrow [])$
		Internal MEAN.
2EEE0	STATSTDEV	$( \rightarrow \% )$
		$( \rightarrow [])$
		Internal SDEV.
2EEE1	STATTOT	$( \rightarrow \% )$
		$( \rightarrow [])$
		Internal TOT.
2EEE2	STATVAR	$( \rightarrow \% )$
		$( \rightarrow [] )$
		Internal VAR.

# **Chapter 11 Composite Objects**

Composite objects hold other objects inside them. In contrast to arrays, different types of objects can be part of the same composite. We have already encountered composite objects in the Introduction, when we used a secondary to group several commands into a single object.

All composites are similar in structure: they start with a word which varies depending on the kind of object, and end with the word SEMI.

Besides secondaries, other composite objects are lists, symbolic objects (described in Chapter 14) and unit objects (described in Chapter 13).

You can create a list by starting it with {, and ending it with }. Inside, put as many objects as you wish, of any kind. Secondaries are delimited with :: and ;.

To concatenate two composites, put them in the stack and use &COMP. To add just one object to the head (beginning) or tail (end) of a composite, first put the composite in the stack, then the object, and call >HCOMP or >TCOMP, respectively. To get the length of the composite (the number of objects, as a bint), just put the composite in level one and use the command LENCOMP. To explode the composite into all its objects and a count (like the User RPL command  $OBJ \rightarrow$ ), use INNERCOMP. The only difference is that the number of objects is returned as a bint. To get one object of a composite, put the composite in level two, the object's position in level one (as a bint, naturally), and run NTHELCOMP. If the number were out of range, you would get a FALSE, otherwise the object and TRUE. NTHCOMPDROP is the above entry, followed by DROP. And to get part of a composite, use the function SUBCOMP. This function takes in level three the composite, in level two the start position (guess what? a bint) and in level one the end position (from now on, unless otherwise noted, all numeric arguments are bints). You will get a composite (of the same type, obviously) with the elements between the start and end positions, inclusive. This function checks if the numbers are not out of range. If they are, a null composite (an empty composite) is returned. The same happens if the end position is greater than the start position.

Other commands are listed in the reference section below.

# 11.1.1 General Operations

Addr.	Name	Description
0521F	&COMP	( comp comp' $ ightarrow$ comp'')
		Concatenates two composites.
052FA	>TCOMP	( comp ob $ o$ comp+ob )
		Adds ob to tail (end) of composite.
052C6	>HCOMP	( comp ob $ ightarrow$ ob+comp )
		Adds ob to head (beginning) of composite.
39C8B	(SWAP>HCOMP)	( ob comp $\rightarrow$ ob+comp )
		Does SWAP then >HCOMP.
05089	CARCOMP	$( comp \rightarrow ob\_head )$
		$(comp_null \rightarrow comp_null)$
		Returns first object of the composite, or a null
		composite if the argument is a null composite.
361C6	?CARCOMP	( comp T $ ightarrow$ ob )
		( comp F $ ightarrow$ comp )
		If the flag is TRUE, does CARCOMP.
05153	CDRCOMP	$(comp \rightarrow comp-ob\_head)$
		$(comp_null \rightarrow comp_null)$
		Returns the composite minus its first object,
		or a null composite if the argument is a null
		composite.
2825E	(2NELCOMPDROP)	$(comp \rightarrow ob2)$
		Gets the second element of composite.
2BC006	^LASTCOMP	$(comp \rightarrow ob)$
		Gets the last element of composite. Does DU-
		PLENCOMP then NTHCOMPDROP.
0567B	LENCOMP	$(\text{comp} \rightarrow \text{#n})$
		Returns length of composite (number of ob-
		jects).
3627A	DUPLENCOMP	$(\text{comp} \rightarrow \text{comp } \# n)$
05555		Does DUP then LENCOMP.
055B7	NULLCOMP?	$(\text{comp} \rightarrow \text{flag})$
26066	DIIDIIII I COMO	If the composite is empty, returns TRUE.
36266	DUPNULLCOMP?	( comp → comp flag )
		Does DUP then NULLCOMP?.

Addr.	Name	Description
056B6	NTHELCOMP	( comp $\#i \to ob T$ )
		( comp $\#i \to F$ )
		Returns specified element of composite and
		TRUE, or just FALSE if it could not be found.
35BC3	NTHCOMPDROP	( comp $\#i \to ob$ )
		Does NTHELCOMP then DROP.
35D58	NTHCOMDDUP	( comp $\#i \to ob ob$ )
		Does NTHCOMPDROP then DUP.
376EE	POSCOMP	( comp ob pred $ ightarrow$ #i )
		( comp ob pred $ ightarrow$ #0 )
		(eg: pred = ' %<)
		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:
		:: { %1 %2 %3 %-4 %-5 %6 %7 } %0
		' %< POSCOMP ;
3776B	EQUALPOSCOMP	( comp ob $ ightarrow$ #pos )
		(comp ob $\rightarrow$ #0)
		POSCOMP with EQUAL as test.
37784	NTHOF	( ob comp $ ightarrow$ #i )
		( ob comp $\rightarrow$ #0 )
		Does SWAP then EQUALPOSCOMP.
0FD006	^ListPos	$( ob \{ \} \rightarrow \#i / \#0 )$
		Equivalent to NTHOF, but faster. However, it
		only works for lists.
37752	#=POSCOMP	$(comp # \rightarrow #i)$
		$(\text{comp } \# \rightarrow \# 0)$
		POSCOMP with #= as test.
05821	SUBCOMP	( comp $\#m \ \#n \rightarrow comp'$ )
		Returns a sub-composite. Makes all index
		checks first.
376B7	matchob?	( ob comp $\rightarrow$ T )
		$( ob comp \rightarrow ob F )$
		Returns TRUE if ob is EQUAL to any element
		of the composite.

Addr.	Name	Description
371B3	Embedded?	( ob1 ob2 $ ightarrow$ flag )
		Returns TRUE if ob2 is embedded in, or is the
		same as, ob1. Otherwise returns FALSE.
37798	Find1stTrue	( comp test $ ightarrow$ ob T )
		( comp test $ ightarrow$ F )
		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 pro-
		gram below returns %-4 and TRUE.
		:: { %1 %2 %2 %-4 %-5 %6 } ' %0<
27705	Lookun	Find1stTrue ;
377C5	Lookup	( ob test comp $ ightarrow$ nextob T ) ( ob test comp $ ightarrow$ ob F )
		Tests every odd element $(1,3,)$ in the com-
		posite. 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 ;
377DE	Lookup.1	( ob test $ ightarrow$ nextob T )
		( ob test $ ightarrow$ ob F )
		Return Stack:
		$(comp \rightarrow )$
		Lookup with the composite already pushed
		(with $>R$ ) onto the runstream. Called by
0.000	1	Lookup.
37829	EQLookup	( ob comp → nextob T )
		$( ob comp \rightarrow ob F )$
		Lookup with EQ as test.

Addr.	Name	Description
37B54	NEXTCOMPOB	( comp #ofs $\rightarrow$ comp #ofs' ob T )
		( comp #ofs $ ightarrow$ 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.

### 11.1.2 Building

There are also shortcut words to build lists and secondaries, with specified number of objects, described in the sections below.

Addr.	Name	Description
05459	{ }N	( obnob1 $\#$ n $\rightarrow$ { obnob1 } )
05445	::N	( oblobn $\#n \rightarrow :: oblobn ; )$
0546D	SYMBN	( oblobn $\#n \to \text{symb}$ )
		Build a symbolic object.
05481	EXTN	( oblobn $\#n \rightarrow u$ )
		Builds a unit object.
293F8	$P\{\}N$	( oblobn $\#n \rightarrow \{\}$ )
		Build list with possible garbage collection.

### 11.1.3 Exploding

Addr.	Name	Description
054AF	INNERCOMP	( comp $\rightarrow$ obnob1 $\#$ n )
3622A	DUPINCOMP	( comp $\rightarrow$ comp obnob1 $\#$ n )
3623E	SWAPINCOMP	( comp obj $\rightarrow$ obj obnob1 $\#$ n )
35BAF	INCOMPDROP	$(comp \rightarrow obnob1)$
35C68	INNERDUP	( comp $\rightarrow$ obnob1 $\#$ n $\#$ n )
2F0EC	ICMPDRPRTDRP	( comp $\rightarrow$ obnob4 ob2 ob1 )
		Does INCOMPDROP then ROTDROP.
3BADA	(INNERCOMP>%)	( comp $\rightarrow$ obnob1 %n )
366E9	INNER#1=	( comp $\rightarrow$ obnob1 flag )

Addr.	Name	Description
157006	^SYMBINCOMP	( symb $ ightarrow$ ob1 obN $\#$ n )
		( ob $ ightarrow$ ob #1 )
		( $\{\} \rightarrow \{\} \#1$ )
		Explodes symbolic object into meta. Other ob-
		jects are converted into one-object metas by
		pushing #1 into the stack.
12A006	^2SYMBINCOMP	( ob1 ob2 $ ightarrow$ meta1 meta2 )
		Does ^SYMBINCOMP for 2 objects.
158006	^CKINNERCOMP	( $\{\} \rightarrow \text{ob1} \dots \text{obN \#n}$ )
		( ob $\rightarrow$ ob $\#1$ )
		Explodes a list into a meta object. Other ob-
		jects are converted into one-object metas by
		pushing #1 into the stack.

#### 11.1.4 Lists

Addr. Name Description	
$055E9  \text{NULL}\{\} \qquad ( \rightarrow \{\} )$	
Pushes a null list to the stack.	
36ABD DUPNULL $\{\}$ ? ( $\{\} ightarrow \{\}$ flag)	
159006 $^{DUPCKLEN\{\}}$ ( $\{\}  ightarrow \{\} \ \#n$ )	
( ob $ ightarrow$ ob #1 )	
Return length of list, or 1 for non-	lists.
29D18 ONE $\{\}$ N (ob $ ightarrow$ {ob $\}$ )	
36202 TWO{}N (oblob2 $\rightarrow$ {oblob2})	
36216 THREE $\{\}$ N ( obl ob2 ob3 $ ightarrow$ { obl ob2	ob3 } )
361EE $\#1-\{\}N$ (oblobn $\#n+1 \rightarrow \{\}$ )	
2B42A PUTLIST ( ob #i $\{\}  ightarrow \{\}$ ' )	
Replaces object at specified position	on. Assumes
valid #i.	
2FC006 ^INSERT $\{\}$ N ( $\{\}$ ob $\# \rightarrow \{\}$ ')	
Insert object into list at given posi-	tion. The po-
sition must be $<$ than length of th	ne list. If the
position is zero, >TCOMP is used.	
2FB006 ^NEXTPext ( list $ ightarrow$ list1 list2 )	
Extract in list2 all occurrances of t	the 1st object
of list, the remaining objects are st	ored in list1.
list1 = list-list2.	

Addr.	Name	Description
2FD006	^COMPRIMext	( {} → {}' )
		Suppress multiple occurrances in the list.
15A006	^CKCARCOMP	( $\{\} \rightarrow ob1$ )
		$( ob \rightarrow ob )$
		Returns first element for lists, or object itself if
		it is not a list.
2EF5A	apndvarlst	$( \{ \} \text{ ob } \rightarrow \{ \} ' )$
		Appends ob to list if not already there.
0FE006	^AppendList	$( \{ \} ob \to \{ \} ' )$
		Equivalent to appndvarlst, but faster.
4EB006	^prepvarlist	( $\{\}$ ob $\rightarrow$ $\{\}$ ')
		Adds ob at the beginning of the list if not
		present. If ob is in list, move ob to the begin-
		ning of list.
100006	^SortList	( L pred $\rightarrow$ L' )
		Sorts list according to give predicate. Pred is
		a program that tests two elements and returns
		FALSE if the first is to appear earlier than the
		second. To sort in numerical order, for exam-
	_	ple, the predicate would be a > test.
28A006	^PIext	$( \{\} \rightarrow ob )$
		Returns the product of all elements of the list.
25ED3	EqList?	$( ob \rightarrow )$
		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.

#### 11.1.5 Secondaries

Addr.	Name	Description
055FD	NULL::	$( \rightarrow :: ; )$
		Returns null secondary.
37073	Ob>Seco	( ob $\rightarrow$ :: ob ; )
		Does one then $::$ N.
3705A	?Ob>Seco	( ob $\rightarrow$ :: ob ; )
		If the object is not a secondary, does Ob>Seco.
37087	20b>Seco	( obl ob2 $ ightarrow$ :: obl ob2 ; )
		Does TWO then :: N.

Addr.	Name	Description
3631A	::NEVAL	( oblobn $\#n \rightarrow ?$ )
		Does::Nthen EVAL

# Chapter 12 Meta Objects

A meta object (or just meta for short) is a collection of n objects and their count (as a bint). A meta object can be considered as another representation of a composite object. The word INNERCOMP will explode any composite into a meta object. The opposite transformation is done by several different words, depending on the kind of composite desired. The available words are listed in section 11.1.2.

Note that a single zero is an (empty) meta object, the null meta object.

It is possible to do several stack operations which treat meta objects as a single object. Generally, the name of these stack operations are in lower case. However, some words have totally misleading names, because their functions are not always used in relation to meta objects, and they were named with their other purpose in mind.

There exist also the user meta objects, which are like meta objects, but the count is represented as a real number and not as a bint. These are not very common, though.

#### 12.1 Reference

#### 12.1.1 Stack Functions

Addr.	Name	Description
0326E	NDROP	( meta $ ightarrow$ )
		Should be called drop.
37032	DROPNDROP	( meta ob $ ightarrow$ )
		Should be called DROPdrop.
35FB0	#1+NDROP	( ob meta $ ightarrow$ )
		Should be called dropDROP.
		aka: N+1DROP

Addr.	Name	Description
28211	NDROPFALSE	( meta $ ightarrow$ F )
		Should be called dropFALSE.
391006	^NDROPZERO	( obnob1 $\#n \rightarrow \#0$ )
		Replace Meta object with empty Meta object.
		Should be called dropZERO.
29A5D	psh	( meta1 meta2 $ ightarrow$ meta2 meta1 )
		Should be called swap.
29A8F	roll2ND	( meta1 meta2 meta3 $ ightarrow$ meta2 meta3
		metal )
		Should be called rot.
29B12	unroll2ND	( metal meta2 meta3 $ ightarrow$ meta3 meta1
		meta2 )
		Should be called unrot.
3695A	SWAPUnNDROP	( meta1 meta2 $ ightarrow$ meta2 )
		Should be called swapdrop.
36FA6	metaROTDUP	( metal meta2 meta3 $ ightarrow$ meta2 meta3
		metal metal )
		Should be called rotdup.

# **12.1.2 Combining Functions**

Addr.	Name	Description
296A7	top&	( meta1 meta2 $\rightarrow$ meta1&meta2 )
2973B	pshtop&	( metal meta2 $ ightarrow$ meta2&meta1 )
36FBA	ROTUntop&	( meta1 meta2 meta3 $\rightarrow$ meta2 meta3&meta1 )
36FCE	roll2top&	( meta1 meta2 meta3 $\rightarrow$ meta3 meta1&meta2 )
		aka: rolltwotop&
2963E	psh&	( meta1 meta2 meta3 $\rightarrow$ meta1&meta3 meta2 )

### 12.1.3 Meta and Object Operations

Addr.	Name	Description
3592B	SWAP#1+	( $\#$ ob $\rightarrow$ ob $\#+1$ )
		aka: SWP1+
34431	DUP#1+PICK	( $n1 \# n \rightarrow n1 \# n n$ )
34504	get1	( ob meta $ ightarrow$ meta ob )
36147	OVER#2+UNROL	( meta ob $ ightarrow$ ob meta )
29693	psh1top&	( meta ob $ ightarrow$ ob&meta )
28071	pull	( $meta\&ob \rightarrow meta \ ob$ )
		aka: #1-SWAP
28085	pullrev	( ob&meta $\rightarrow$ meta ob )
29821	psh1&	( meta1 meta2 ob $\rightarrow$ ob&meta1 meta2 )
298C0	psh1&rev	( metal meta2 ob $ ightarrow$ ob&meta1 meta2 )
2F193	UobROT	( ob meta1 meta2 $ ightarrow$ meta1 meta2 ob )
29754	pullpsh1&	( metal meta2&ob $\rightarrow$ ob&meta1 meta2 )
406006	^addt0meta	( metal&ob meta2 $ ightarrow$ meta1 meta2 )
		Removes the last object of meta1.
29972	pshzer	( meta $\rightarrow$ #0 meta )
36946	SWAPUnDROP	( ob meta $ ightarrow$ meta )
2F38E	xnsgeneral	( meta $\rightarrow$ LAM3&meta&LAM1 )
		Uses contents of LAM1 and LAM3.
2F38F	xsngeneral	( meta $\rightarrow$ meta&LAM3&LAM1 )
		Uses contents of LAM1 and LAM3.

# 12.1.4 Other Operations

Addr.	Name	Description
3760D	SubMetaOb	( meta $\#$ start $\#$ end $\to$ meta' )
		Gets a sub-meta. Does range checks.

Addr.	Name	Description
37685	SubMetaOb1	( ob1obiobn #n #i #n #i →
		ob1obi #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
33F006	^submeta	( meta #begin #end $ ightarrow$ meta' )
		Extracts submeta from a meta.
2F356	metatail	( oblobn-iobn #i #n+1 $\rightarrow$
		oblobobn-i #n-i obn-i+1obn
		#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
385006	^metasplit	( meta $\#i \to meta1 meta2$ )
		Split a meta in 2 metas at position i. meta1
		will contain #i elements meta2 will contain
		#n-i elements.
39F006	^metaEQUAL?	( meta2 meta1 $ ightarrow$ meta2 meta1 flag )
		Test equality of 2 metas.
3BF006	^EQUALPOSMETA	( Meta ob $\rightarrow$ Meta ob $\#$ pos )
		Returns last occurrence of ob in Meta. If a
		component of meta is a list/symb then search
		if ob is embedded in this component of meta.
3C0006	^EQUALPOS2META	( Meta2 Meta1 ob $\rightarrow$ Meta2 Meta1 ob
		#pos )
		Returns last occurrence of ob in Meta1 or in
		Meta2. #pos is >0 if in meta2, is <0 if in
	<b>^</b>	meta1 (#pos=MINUSONE-#).
198006	^METAINT?	( Meta → Meta flag )
		Tests if Meta is an integer.

Addr.	Name	Description								
199006	^METAPOSINT?	<ul> <li>( Meta → Meta flag )</li> <li>Tests if Meta is a positive integer smaller than Zsmall.</li> </ul>								

# Chapter 13 Unit Objects

Units are another kind of composite objects. It is not really difficult to include one in the program, it is just laborious.

Units start with UNIT and end with : Inside, there are commands to define the unit. The best way to understand how a unit is reprsented is by disassembling it. The unit object 9.8\_m/s^2 can be created using the code below:

As you can see, creating units is done in a reverse polish way using the words um^, um\*, um/ and ump. The meaning of the first three ones is easy to guess. The last is used to create prefix operators (kilo, mega, mili, etc.). First enter the prefix as a character or string, and then the unit name (all operations take unit names as characters or strings). Run ump and the prefixed unit is created. Then call the other functions as needed. To end a unit, use umend, which joins the number (entered first) to the unit part. The code above could be made shorter if built-in characters and strings (listed on Chapter 6) were used.

Since units are composite objects, you can use, for example, INNERCOMP to explode a unit into a meta object. You can also create a unit from a meta object (see Chapter 12), using EXTN. The program below, for example, adds the unit m/s to the number in the stack:

Note that the um words, when executed, just put themselves in the stack.

Several operations can be done with units. The complete list is given below. The most important are UM+, UM-, UM\*, UM/ and UFACT, whose meanings are obvious; UMCONV, which works like user word CONVERT; UMSI, equivalent to UBASE and U>nbr, which returns the numeric part of a unit.

#### 13.1 Reference

#### 13.1.1 Creating Units

Addr.	Name	Description
2D74F	um*	* marker
2D759	um/	/ marker
2D763	um^	^ marker
2D76D	umP	Char prefix operator
2D777	umEND	Unit end operator
05481	EXTN	( oblobn $\#n \rightarrow u$ )
		Builds a unit object.

# 13.1.2 General Functions

Addr.	Name	Description					
2F099	U>NCQ	( $u \rightarrow n\%$ cf%% qhxs )					
		Returns the number, conversion factor to base units					
		and a vector in the form:					
		[kgmAsKcdmolrsr?]					
		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 0 ]					
		since it is equivalent to 1_kg*m/s^2					
2F07A	UM>U	( % u → u' )					
		Replaces number part of unit.					
2F08C	UMCONV	( u1 u2 $\rightarrow$ u1' )					
		Change units of unit1 to units of unit2.					
2F090	UMSI	$(u \rightarrow u')$					
		Equivalent to user word UBASE.					
2F095	UMU>	$(u \rightarrow % u')$					
		Returns number and normalized part of unit.					
2F019	$\mathtt{UNIT}{>}\$$	$(u \rightarrow $)$					
		Converts unit to string.					
2F07B	U>nbr	$(u \rightarrow %)$					
		Returns number part of unit.					
2F098	Unbr>U	$(u \% \rightarrow u')$					
		Replaces number part of unit.					
2F09A	TempConv	???					
		Used by UMCONV for the conversion of temperature					
		units.					
25EE4	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.					

### 13.1.3 Arithmetic Functions

Addr.	Name	Description
2F081	UM+	( u u' → u'' )
2F082	UM-	( u u' $\rightarrow$ u'' )
2F080	UM*	( $u~u'~\rightarrow~u''$ )
2F083	UM/	( $u~u'~\rightarrow~u''$ )
2F07D	UM%	( u %percent $\rightarrow$ u' )
2F07E	UM%CH	$(uu' \rightarrow %)$
2F07F	UM%T	$(uu' \rightarrow %)$
2F08F	UMMIN	$(uu' \rightarrow u?)$
2F08E	UMMAX	$(uu' \rightarrow u?)$
2F096	UMXROOT	$(uu' \rightarrow u'')$
2F08A	UMABS	$(u \rightarrow u')$
2F08B	UMCHS	$(u \rightarrow u')$
2F092	UMSQ	$(u \rightarrow u')$
2F093	UMSQRT	$(u \rightarrow u')$
2D949	UMSIGN	$(u \rightarrow u')$
2D95D	UMIP	$(u \rightarrow u')$
2D971	UMFP	$(u \rightarrow u')$
2D985	UMFLOOR	$(u \rightarrow u')$
2D999	UMCEIL	$(u \rightarrow u')$
2D9CB	UMRND	$(u \rightarrow u')$
2D9EE	UMTRC	$(u \rightarrow u')$
2F08D	UMCOS	$(u \rightarrow u')$
2F091	UMSIN	$(u \rightarrow u')$
2F094	UMTAN	$(u \rightarrow u')$

#### 13.1.4 Tests

Addr.	Name	Description
2F087	UM=?	( u u' $\rightarrow$ %flag )
2F07C	UM#?	( u u' $ ightarrow$ %flag )
2F086	UM </td <td>( u u' <math> ightarrow</math> %flag )</td>	( u u' $ ightarrow$ %flag )
2F089	UM>?	( u u' $ ightarrow$ %flag )
2F085	UM <= ?	( u u' $ ightarrow$ %flag )
2F088	<pre>UM&gt;=?</pre>	( u u' $\rightarrow$ %flag )

Addr.	Name	Descr	ipti	on						
2F076	puretemp?	( [] []' $ ightarrow$ [] []' flag )								
		Checks of the two arrays both denote pure temperature units, i.e. if both arrays are equal to								
		[0. 0.]	0.	0.	0.	1.	0.	0.	0.	0.

# Chapter 14 Symbolics

Symbolic objects, or algebraic expressions, are another type of composite objects. Their structure is very similar to the units'. They are delimited by SYMBOL and *i*. Inside, the expression is created in a reverse polish way.

The disassembly of the equation  $R=\frac{V}{I}$  should show how to include a symbolic object in your program, should you need one.

As you have seen, the variables are represented by identifiers, and the functions are the user-accessible ones, whose names are preceded by a lower-case  $\mathbf{x}$  in System RPL.

To create a symbolic object from a meta, you use the SYMBN function. Note that when you include a funcion, you will have to quote it, ie, put the command ' before the command to put it in the stack instead of executing it. Quoting objects will be explained in more detail in section 19.2.

On the HP49, the new CAS contains most entries dealing with symbolics. These entries are described in the CAS part of the book. mainly in Chapters 44 and 45. However, some entries which were available already on the HP48 have been kept for compatibility reasons. These entries are listed below.

86 14. Symbolics

# 14.1 Reference

# 14.1.1 General Operations

Addr.	Name	Description
0546D	SYMBN	( oblobn $\#n \to \text{sym}$ )
2BD8C	(SYMBN:)	ob1obn #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.
286E7	symcomp	$(ob \rightarrow ob')$
		If ob is symbolic, does nothing, otherwise ONE SYMBN.
2F073	SWAPcompSWAP	( ob ob' $ ightarrow$ ob'' ob' )
		Does SWAP symcomp SWAP.
28ACE	(DROP?symcomp)	( %/C%/Z/id/lam ob' $\rightarrow$
		%/C%/Z/id/lam )
		( ob ob' $ ightarrow$ symb )
		Drop ob'. Then, if the object in the stack is
		a real, complex, zint, identifier or lam, does
		nothing. For other objects, calls symcomp to
		create a one-object symbolics.
293A3	(?symcomp)	( $%/C%/Z/id/lam #1 \rightarrow %/C%/Z/id/lam$ )
		( ob $\#1 \to \text{symb}$ )
		$( ob \# \rightarrow symb )$
		If # is BINT1, calls DROP?symcomp. If it is
		any other number, calls SYMBN.
25EA2	CRUNCH	$( ob \rightarrow \% )$
		Internal version of $ ightarrow$ NUM.
2F110	(FINDVARS)	( $\text{sym} \rightarrow \{\}$ )
		Returns a list of the variables of the equation,
		recursing into programs and functions in the
		equation.
462006	^EQUATION?	( ob $ ightarrow$ ob flag )
		Returns TRUE if ob is a symbolic finishing by
		x=.

Addr.	Name	Description
463006	^USERFCN?	( ob → ob flag )  Returns TRUE if ob is a symbolic finishing by xFCNAPPLY.
29CB9	uncrunch	<pre>( → ) Clears numeric results flag (system flag 3) for the next command only. Example: SYMCOLCT = :: uncrunch colct;</pre>
2BCA2	cknumdsptch1	<pre>( sym → symf ) Used by one argument functions to evaluate a symbolic or numeric routine according to nu- meric 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 ;</num></num></sym></pre>
2BB21	sscknum2	<pre>( 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+ = :: sncknum2 Meta+ x+;</num></num></sym></pre>
2BB3A	sncknum2	<pre>( sym % → symf ) Usage: :: sncknum2 <sym> <num> ; In symbolic mode uses cksneval2:. Example: SYM+O = :: sncknum2 Meta+Con x+ ;</num></sym></pre>
2BB53	nscknum2	( % sym → symf ) Usage: :: nscknum2 <sym> <num>; In symbolic mode uses cknseval2:. Example: O+SYM = :: nscknum2 Con+Meta x+;</num></sym>

88 14. Symbolics

## 14.1.2 Other Functions

Addr.	Name	Description
2EF26	SYMSHOW	( sym id/lam $\rightarrow$ symf )
2F2A9	XEQSHOWLS	( $sym\ \{\} \to symf$ )

# 14.1.3 Meta Symbolics Functions

Addr.	Name	Description
29986	pshzerpsharg	( meta $ ightarrow$ M_last M_rest )
		Pushes last sub-expression in meta. If meta is
		a valid expression M_rest will be empty.
3701E	pZpargSWAPUn	( meta $ ightarrow$ M_rest M_last )
		pshzerpsharg then psh.
36FE2	plDRPpZparg	( meta&ob $ ightarrow$ M_last M_rest )
		Drops ob then calls pshzerpsharg.
3F1006	^DIVMETAOBJ	(o1on $\#$ n ob $\rightarrow$ {o1/obon/ob})
		Division of all elements of a meta by ob. Tests
		if $o=1$ .

# **Chapter 15 Graphics Objects (Grobs)**

Graphics objects, or grobs for short, represent images, drawings, etc. If you want to write programs that draw something in the screen, then you must know how to use grobs, because the screen content is actually a grob, and you will have to draw on that grob, or to insert another grob in it.

In the reference section below, there are words for creating, manipulating and displaying graphic objects.

When dealing with graphics, keep two things in mind:

- 1. Several grob operations work directly on the grob without making a copy. So, all pointers to that object in the stack will be modified. You can use the word CKREF to ensure an object is unique. For more information on temporary memory and reference counting, see section 24.1.4. This kind of operation is denominated "bang-type", and the commands normally have an exclamation point to indicate that, like GROB! or GROB! ZERO. These operations also have no error checking, so improper or out-of-range parameters may corrupt memory.
- 2. The best command to place a grob in the display grob is XYGROBDISP. This is because this word checks if the grob to be placed in HARDBUFF would exceed its boundaries, and if necessary HARDBUFF is enlarged so that the grob fits.

#### 15.1.1 Built-in Grobs

Addr.	Name	Description
27AA3	(NULLGROB)	( $ ightarrow$ grob )
		0x0 Null grob
27D3F	CROSSGROB	( $ ightarrow$ grob )
		5x5 Cross cursor ("+")
27D5D	MARKGROB	( $ ightarrow$ grob )
		5x5 Mark symbol ("x")
27D7B	(StdLabelGrob)	21x8 normal menu key
2E25C	(InvLabelGrob)	21x8 inverse menu key
0860B0	~grobAlertIcon	9x9 Alert grob
0870B0	~grobCheckKey	21x8 Check Key menu grob
		A tickmark and "CHK" in a menu grob.

### 15.1.2 Dimensions

Addr.	Name	Description
26085	GROBDIM	( grob $ ightarrow$ #height #width )
25EBB	DUPGROBDIM	( grob $ ightarrow$ grob #height #width )
36C68	GROBDIMw	( grob $ ightarrow$ #width )
2F324	CKGROBFITS	( g1 g2 #n #m $ ightarrow$ g1 g2' #n #m )
		Shrinks g2 if it does not fit in g1.
2F320	CHECKHEIGHT	( grob #height $ ightarrow$ )
		Forces grob (ABUFF/GBUFF) to be at least 64
		rows high.
		9

## 15.1.3 Grob Handling

Addr.	Name	Description
2607B	GROB!	( grob1 grob2 $\#x \#y \rightarrow$ )
		Stores grob1 into grob2. Bang type.
2EFDB	(GROB+)	( grob1 grob2 $ ightarrow$ grob )
		Combines two grobs using bitwise OR. Errors
		when grobs have different sizes.

Addr.	Name	Description
2F342	GROB+#	(flag grob1 grob2 $\#x \#y \to grob'$ ) 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.
26080	GROB!ZERO	( grob $\#x1 \#y1 \#x2 \#y2 \rightarrow \text{grob'}$ ) Blanks a rectangular region of the grob. Bang type.
368E7	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.
2612F	SUBGROB	( grob $\#x1 \#y1 \#x2 \#y2 \rightarrow \text{grob}'$ ) Returns specified portion of grob.
25F0E	XYGROBDISP	( $\#x \#y \text{ grob} \rightarrow$ ) Stores grob in HARDBUFF with upper left corner at ( $\#x,\#y$ ). HARDBUFF is expanded if necessary.
25ED8	GROB>GDISP	( grob $\rightarrow$ ) Stores new graph grob.
260B2	MAKEGROB	( $\#$ height $\#$ width $\rightarrow$ grob ) Creates a blank grob.
2F0DB	MAKEPICT#	( $\#w \#h \rightarrow$ ) Creates blank graph grob. Minimum size is 131x64. Smaller grobs will be automatically resized.
2609E	INVGROB	( grob $\rightarrow$ grob' ) Inverts grob data bits. Bang type.
260E4	PIXON	$( \#x \#y \rightarrow )$ Sets pixel in text grob.
260DF	PIXOFF	( #x #y → ) Clears pixel in text grob.
260EE	PIXON?	( $\#x \#y \rightarrow flag$ ) Is pixel in text grob on?
260DA	PIXON3	( #x #y → ) Sets pixel in graph grob.
260D5	PIXOFF3	<ul> <li>( #x #y → )</li> <li>Clears pixel in graph grob.</li> </ul>

Addr.	Name	Description
260E9	PIXON?3	( $\#x \#y \rightarrow flag$ )
		Is pixel in graph grob on?
280C1	ORDERXY#	( $\#x1 \ \#y1 \ \#x2 \ \#y2 \rightarrow \ \#x1' \ \ \#y1' \ \ \#x2'$
		#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.
280F8	ORDERXY%	( %x1 %y1 %x2 %y2 → %x1' %y1' %x2'
		%y2')
		ORDERXY# with real numbers.
2EF9F	LINEON	$( #x1 #y1 #x2 #y2 \rightarrow )$
		Draws a line in text grob.
2EFA0	LINEOFF	$( #x1 #y1 #x2 #y2 \rightarrow )$
		Clears a line in text grob.
2EFA1	TOGLINE	$( #x1 #y1 #x2 #y2 \rightarrow )$
		Toggles a line in text grob.
2EFA2	LINEON3	( #x1 #y1 #x2 #y2 → )
		Draws a line in graph grob.
2F13F	DRAWLINE#3	$( #x1 #y1 #x2 #y2 \rightarrow )$
		Draws a line in graph grob. x1 <x2 is="" not="" re-<="" td=""></x2>
		quired.
2EFA3	LINEOFF3	( #x1 #y1 #x2 #y2 → )
		Clears a line in graph grob.
2EFA4	TOGLINE3	$( #x1 #y1 #x2 #y2 \rightarrow )$
		Toggles a line in graph grob.
2F382	TOGGLELINE#3	$( #x1 #y1 #x2 #y2 \rightarrow )$
		Toggles line in graph grob. x1 <x2 is="" not="" re-<="" td=""></x2>
		quired.
2F32C	DRAWBOX#	( #x1 #y1 #x2 #y2 → )
		Draws rectangle in graph grob.
2EF03	DOLCD>	$( \rightarrow grob )$
		Returns current display.
2EF04	DO>LCD	$( grob \rightarrow )$
		Grob to display.
0BF007	^GROBADDext	( grob2 grob1 → grob )
		Vertical grob addition. grob2 will be above
		grob1.

# 15.1.4 Greyscale Graphics

Addr.	Name	Description
25592	SubRepl	( grb1 grb2 #x1 #y1 #x2 #y2 #W #H $\rightarrow$ grb1' )
		Replace a part of grb1 with a part of grb2 in REPLACE mode.
25597	SubGor	( grb1 grb2 #x1 #y1 #x2 #y2 #W #H $ ightarrow$
		grb1') Replace a part of grb1 with a part of grb2 in OR mode.
2559C	SubGxor	mode. (grb1 grb2 #x1 #y1 #x2 #y2 #W #H $\rightarrow$ grb1')
		Replace a part of grb1 with a part of rgb2 in XOR mode.
25565	LineW	( grb #x1 #y1 #x2 #y2 $ ightarrow$ grb' ) Draw a white line.
2556F	LineG1	( grb #x1 #y1 #x2 #y2 $\rightarrow$ grb' ) Draw a light grey line.
25574	LineG2	( grb #x1 #y1 #x2 #y2 $\rightarrow$ grb' ) Draw a dark grey line.
2556A	LineB	( grb #x1 #y1 #x2 #y2 $\rightarrow$ grb' ) Draw a black line.
25579	LineXor	( grb #x1 #y1 #x2 #y2 $\rightarrow$ grb' ) XOR a line.
2F218	CircleW	( grb #Cx #Cy #r $\rightarrow$ grb' )  Draw a white circle.
2F216	CircleG1	( grb #Cx #Cy #r $ ightarrow$ grb' )
2F217	CircleG2	Draw a light grey circle.  ( grb #Cx #Cy #r → grb' )
2F215	CircleB	Draw a dark grey circle.  ( grb #Cx #Cy #r → grb' )
2F219	CircleXor	Draw a black circle ( grb $\#Cx \#Cy \#r \rightarrow grb'$ ) XOR a circle.
2557E	Sub	( grb $\#$ x1 $\#$ y1 $\#$ x2 $\#$ y2 $ o$ grb' flag )
		Get a part of a grob.
25583	Repl	( grb1 grb2 $\#x \#y \rightarrow grb1'$ ) Copy grb2 into grb1 in REPLACE mode.

Addr.	Name	Description
25588	Gor	( grb1 grb2 $\#x \#y \rightarrow grb1'$ )
		Copy grb2 into grb1 in OR mode.
2558D	Gxor	( grb1 grb2 $\#x \#y \rightarrow grb1'$ )
		Copy grb2 into grb1 in XOR mode.
255A1	Grey?	( grob $\rightarrow$ flag )
		Is grob a Greyscale Grob?
255B0	ScrollVGrob	( grb #W #X #Yd #Ys #h $ ightarrow$ grb' )
		Scroll up and down a portion of a graphical object.
255BA	PixonW	( grb $\#x \#y \rightarrow grb'$ )
		Make a pixel white.
255C4	PixonG1	( grb $\#x \#y \rightarrow grb'$ )
		Make a pixel light grey.
255C9	PixonG2	( grb $\#x \#y \rightarrow grb'$ )
		Make a pixel dark grey.
255BF	PixonB	( grb $\#x \#y \rightarrow grb'$ )
		Make a pixel black.
255CE	PixonXor	( grb $\#x \#y \rightarrow grb'$ )
		Apply XOR to a pixel.
255D3	FBoxW	( grb $\#$ x1 $\#$ y1 $\#$ x2 $\#$ y2 $\rightarrow$ grb' )
		Make a white filled rectangle.
255D3	FBoxG1	( grb $\#$ x1 $\#$ y1 $\#$ x2 $\#$ y2 $\rightarrow$ grb' )
		Make a light grey filled rectangle.
255D8	FBoxG2	( grb $\#$ x1 $\#$ y1 $\#$ x2 $\#$ y2 $ o$ grb' )
		Make a dark grey filled rectangle.
255DD	FBoxB	( grb $\#$ x1 $\#$ y1 $\#$ x2 $\#$ y2 $ o$ grb' )
		Make a black filled rectangle.
255E2	FBoxXor	( grb $\#$ x1 $\#$ y1 $\#$ x2 $\#$ y2 $ o$ grb' )
		Apply XOR to a filled rectangle.
255E7	LBoxW	( grb $\#$ x1 $\#$ y1 $\#$ x2 $\#$ y2 $ o$ grb' )
		Draw a white rectangle.
255EC	LBoxG1	( grb $\#$ x1 $\#$ y1 $\#$ x2 $\#$ y2 $ o$ grb' )
		Draw a light grey rectangle.
255F1	LBoxG2	( grb $\#$ x1 $\#$ y1 $\#$ x2 $\#$ y2 $ o$ grb' )
		Draw a dark grey rectangle.
255F6	LBoxB	( grb $\#$ x1 $\#$ y1 $\#$ x2 $\#$ y2 $ o$ grb' )
		Draw a black rectangle.
255FB	LBoxXor	( grb $\#x1 \ \#y1 \ \#x2 \ \#y2 \rightarrow grb'$ )
		Apply XOR to a rectangle.

Addr.	Name	Description
2F21B	ToGray	$( grb \rightarrow grb'/grb )$
		Convert a B&W grob to Greyscale.
2F21A	Dither	( grb $ ightarrow$ grb'/grb )
		Convert a greyscale grob to B&W
255B5	Distance	$( \#\Delta x \#\Delta y \rightarrow \#SQRT(\Delta x^2+\Delta y^2) )$
		Compute the distance between two points.

# 15.1.5 Creating Menu Label Grobs

Addr.	Name	Description
2E166	MakeStdLabel	$( \ \$ \rightarrow \text{grob} \ )$
		Makes standard menu label.
2E189	MakeBoxLabel	( $\$ \rightarrow \texttt{grob}$ )
		Makes label with a box.
2E1EB	MakeDirLabel	( $\$ \rightarrow \texttt{grob}$ )
		Makes directory label.
2E24D	MakeInvLabel	( $\$ \rightarrow \texttt{grob}$ )
		Makes inverse label.
25E7F	Box/StdLabel	( $\$$ flag $\rightarrow$ grob )
		If TRUE makes box label, otherwise makes stan-
		dard label.
25F01	Std/BoxLabel	( $\$$ flag $ o$ grob )
		If TRUE makes standard label, otherwise makes
		box label.
25E80	Box/StdLbl:	$(  ightarrow  ext{grob})$
		Does Box/StdLabel with the next two objects
		from the stream.
		<pre>Usage: :: Box/StdLbl: \$ <test> ;</test></pre>
2E0D5	Grob>Menu	( $\#$ col grob $ o$ )
		Displays grob as menu label.
2E0F3	Str>Menu	$( \#col $ \rightarrow )$
		Displays string as menu label.
2E11B	Id>Menu	( $\#$ col id $\rightarrow$ )
		Displays id as menu label.
2E107	Seco>Menu	( $\#$ col :: $\rightarrow$ )
		Does EVAL then DoLabel.

Addr.	Name	Description
25886	DoLabel	( $\#$ col ob $\rightarrow$ )
		If ob is of one of the supported types, displays a menu label. If not, generates a "Bad Argument Type" error.

# 15.1.6 Converting Strings to Grobs

Addr.	Name	Description
25F7C	\$>GROB	( $\$ \rightarrow \text{grob}$ )
		Makes grob of the string using the system
		font. Linefeed does <i>not</i> make new line.
25F86	\$>GROBCR	$(\$ \rightarrow grob)$
		Makes grob of the string using the system
		font. Linefeed <i>does</i> make new line.
25F81	\$>grob	$(\$ \to grob)$
05=05	#. 1 an	
25F8B	\$>grobCR	
0.550.00.3	(~ d>	
051083	(\$>groburgrob)	- · · · · · · · · · · · · · · ·
		, ,
25F24	ртснт¢з√6	· ·
231 24	KIGIIIQJXO	
		5
25FEF	CENTER\$3x5	
	·	
		font) and embeds it at specified position (#x,
		#y). The grob is centered around #x and
		the to is put at #y. #w represents the max-
		imum width of the grob created. If the text
		is wider, it is truncated. Bangtype.
25F8B 05F0B3 25F24	<pre>\$&gt;grob  \$&gt;grobCR  (~\$&gt;grobOrGROB)  RIGHT\$3x6  CENTER\$3x5</pre>	Makes grob of the string using the minifont. Linefeed does not make new line.  (\$ → grob\$)  Makes grob of the string using the minifont. Linefeed does make new line.  (\$ → grob\$)  Converts string to a grob using either the current font or the minifont, depending on system flag 90.  (\$ #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.  (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

Addr.	Name	Description
2E2AA	MakeLabel	( \$ #w #x grob → grob' ) Inserts \$ into grob using CENTER\$3x5 with
25FF9	LEFT\$3x5	y=5. ( grob #x #y \$ #w → grob') Like CENTER\$3x5, but the left corner of the text is positioned at #x.
26071	ERASE&LEFT\$3x5	( grob #x #y \$ #w → grob' ) Like LEFT\$3x5, but erase background first.
26008	LEFT\$3x5Arrow	( grob #x #y \$ #w → grob' ) Like LEFT\$3x5, but if the text does not fit, replace the last character by character 31 (dots) to show that the text was truncated.
2601C	LEFT\$3x5CR	(grob #x #y \$ #w #h → grob')  Like LEFT\$3x5, but newlines in the strings are interpreted and start new lines. Note the additional argument #h for the maximum height of the text grob.
26012	LEFT\$3x5CRArrow	( grob #x #y \$ #w #h → grob' ) Like LEFT\$3x5CR, but show truncation with arrows.
25FF4	CENTER\$5x7	( grob $\#x \#y \$ \#w \rightarrow \text{grob'}$ ) Same as CENTER\$3x5, but using system font.
25FFE	LEFT\$5x7	( grob #x #y \$ #w $\rightarrow$ grob' ) Like CENTER\$5x7, but the left corner of the text is positioned at #x.
2606C	ERASE&LEFT\$5x7	( grob $\#x \#y \$ \#w \rightarrow grob'$ ) Like LEFT\$5x7, but erase background first.
26003	LEFT\$5x7Arrow	( grob #x #y \$ #w → grob') Like LEFT\$5x7, but if the text has to be truncated, replace the last character with character 31 (arrow).
26017	LEFT\$5x7CR	( grob $\#x \#y \$ \#w \rightarrow grob'$ ) Like LEFT\$5x7, but interpret newlines.
2600D	LEFT\$5x7CRArrow	( grob #x #y \$ #w $\rightarrow$ grob' ) Like LEFT\$5x7CR, but show truncation with arrows.

## 15.1.7 Creating Grobs from Other Objects

Addr.	Name	Description
019004	^EQW3GROB	( ob $\rightarrow$ ext grob #0 )
		( ob $\rightarrow$ #2 )
01A004	^EQW3GROBStk	( ob $ ightarrow$ ext grob #0 )
		( ob $\rightarrow$ #2 )
01F004	^EQW3GROBmini	( ob $ ightarrow$ ext grob #0 )
		( ob $\rightarrow$ #2 )
01E004	^EQW3GROBsys	( ob $ ightarrow$ ext grob #0 )
		( ob $\rightarrow$ #2 )
0BE007	^XGROBext	( ob $ ightarrow$ grob )
		Convert object to a grob.
0C0007	^DISPLAYext	( grob ob $ ightarrow$ grob' )
		Adds ob to grob after converting it to a grob.

# **Chapter 16 Library and Backup Objects**

Libraries are very complex objects that hold a collection of commands. Some of these commands are named and accessible to the user, but some have no names, and so are "hidden". Backup objects are used by the HP49 to store the contents of the entire HOME directory and restore it later. The integrity of both objects can be verified because both have a CRC code attached to them.

A rompointer (sometimes called XLIB name) is a pointer to a command in a library. The only way to access a unnamed command in a library is through a rompointer. They hold the number (often called id) of the library and the number of the command.

To insert a rompointer in your program, use the following structure: ROMPTR <lib> <cmd>, where <lib> is the number of the library, and <cmd> is the number of the command. Both numbers are specified in hexadecimal form. Rompointers are always automatically executed (like identifiers), so you have to quote them (see section 19.2) if you want one in the stack.

#### 16.1 Reference

## 16.1.1 Port Operations

Addr.	Name	Description			
25EEB	NEXTLIBBAK	( $\#$ addr $\rightarrow$ backup/library $\#$ nextaddr )			
Gets next library or backup.					

## 16.1.2 Rompointers

Addr.	Name	Description
07E50	#>ROMPTR	( $\#$ lib $\#$ cmd $\to$ ROMPTR )
		Creates rompointer.
08CCC	ROMPTR>#	( ROMPTR $ ightarrow$ #lib #cmd )
		Splits rompointer.
07E99	ROMPTR@	( ROMPTR $\rightarrow$ ob T )
		( ROMPTR $ ightarrow$ F )
		Recalls contents of rompointer.
35C40	DUPROMPTR@	( ROMPTR $\rightarrow$ ROMPTR ob T )
		( ROMPTR $ ightarrow$ ROMPTR F )
		Does DUP then ROMPTR@.
35A88	?>ROMPTR	$( ob \rightarrow ob' )$
		If ROM-WORD? and TYPECOL? then RPL@.
35AAB	?ROMPTR>	$( ob \rightarrow ob' )$
		If TYPEROMP? and content exists INHARDROM?
		then return contents.
35BFF	RESOROMP	$( \rightarrow \text{ ob })$
		Recalls contents of next object in the runstream
		(which must be a rompointer).
34FCD	ROM-WORD?	$( ob \rightarrow flag )$
34FC0	DUPROM-WORD?	( ob $\rightarrow$ ob flag )

## 16.1.3 Libraries

Addr.	Name	Description
07709	TOSRRP	( ♯ → )
		Attaches library to HOME directory.
076AE	OFFSRRP	$( \ \# \  o \ )$
		Detaches library from HOME directory.
2F2A7	XEQSETLIB	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
		Internal ATTACH.
07638	SETHASH	( hxs $\#$ libnum $\rightarrow$ )
		Buggy?

# 16.1.4 Backup Objects

Addr.	Name	Description
081D9	BAKNAME	( bak $\rightarrow$ id T )
		Returns backup's name
0905F	BAK>0B	( bak $ ightarrow$ ob )
		Gets backup object.

# Part II

# General System RPL Entries

# **Chapter 17 Stack Operations**

In System RPL, using the stack is almost the same as in User RPL. The basic operations are the same, except for little changes in the name: DUP, 2DUP (equivalent to User RPL's DUP2), NDUP (DUPN), DROP, 2DROP (DROP2), NDROP (DROPN), OVER, PICK, SWAP, ROLL, UNROLL (ROLLD), ROT, UNROT and DEPTH.

All commands that require or return a numeric argument use bints and not real numbers, unless otherwise noted.

There are many commands that do two or even three operations in sequence. They are listed in the reference section. The table below lists some useful combinations in a nice form:

	DUP	DROP	SWAP	OVER	ROT	UNROT
DUP	DUPDUP	DROPDUP	SWAPDUP	OVERDUP	ROTDUP	UNROTDUP
DROP	_	2DROP	SWAPDROP	_	ROTDROP	UNROTDROP
SWAP	_	DROPSWAP	_	OVERSWAP	ROTSWAP	UNROTSWAP
OVER	DUPDUP	DROPOVER	SWAPOVER	2DUP	ROTOVER	UNROTOVER
ROT	DUPROT	DROPROT	SWAPROT	-	UNROT	_
UNROT	DUPUNROT	_	ROTSWAP	OVERUNROT	_	ROT
SWAPDROP	_	${\tt DROPSWAPDROP}$	_	DROPDUP	DROPSWAP	UNROTSWAPDRO
DROPDUP	_	_	SWAPDROPDUP	-	_	_
DROPSWAP	_	_	SWAPDROPSWAP	_	ROTDROPSWAP	SWAPDROP
2DROP	_	3DROP	_	-	ROT2DROP	UNROT2DROP
2DUP	_	_	SWAP2DUP	_	ROT2DUP	_
3PICK	DUP3PICK	_	SWAP3PICK	OVERDUP	_	_
4PICK	_	-	SWAP4PICK	-	_	-
5PICK	_	-	_	OVER5PICK	_	-
4ROLL	_	_	SWAP4ROLL	_	_	_
4UNROLL	${\tt DUP4UNROLL}$	-	_	-	_	-
ROT2DROP	-	=	ROTROT2DROP	SWAPDROP	ROTROT2DROP	=

## 17.1 Reference

In this section, the numbers 1, 2... n are used to represent different objects, not necessarily any kind of number.

Addr.	Name	Description
03188	DUP	( ob $\rightarrow$ ob ob )
35CE0	DUPDUP	( ob $ ightarrow$ ob ob ob )
2D5006	^3DUP	$(\ \ 3\ \ 2\ \ 1\ \ \rightarrow\ \ 3\ \ 2\ \ 1\ \ 3\ \ 2\ \ 1\ \ )$
28143	NDUPN	( ob $\#n \rightarrow obob \ \#n$ )
		( ob $\#0 \rightarrow \#0$ )
35FF3	DUPROT	$(12 \rightarrow 221)$
3457F	DUPUNROT	$(12 \rightarrow 212)$
		aka: SWAPOVER
36133	DUPROLL	$(1n \#n \rightarrow 1 3n \#n 2)$
3432C	DUP4UNROLL	$(123 \rightarrow 3123)$
3611F	DUPPICK	( $n1 \# n \rightarrow n1 \# n n-1$ )
35D30	DUP3PICK	$(12 \rightarrow 1221)$
		aka: 2DUPSWAP
34431	DUP#1+PICK	( $n1 \# n \rightarrow n1 \# n n$ )
031AC	2DUP	$(12 \rightarrow 1212)$
35D30	2DUPSWAP	$(12 \rightarrow 1221)$
		aka: DUP3PICK
36CA4	2DUP5ROLL	$(123 \rightarrow 23231)$
031D9	NDUP	$(1n \#n \rightarrow 1n 1n)$
03244	DROP	$(1 \rightarrow)$
357CE	DROPDUP	$(12 \rightarrow 11)$
37032	DROPNDROP	$(1n \#n ob \rightarrow )$
35733	DROPSWAP	$(123 \rightarrow 21)$
3574D	DROPSWAPDROP	$( 1 2 3 \rightarrow 2 )$
		aka: ROT2DROP, XYZ>Y
36007	DROPROT	$(1234 \rightarrow 231)$
3606B	DROPOVER	$(123 \rightarrow 121)$
03258	2DROP	$(12 \rightarrow)$
341D2	3DROP	$(123\rightarrow)$
		aka: XYZ>
341D7	4DROP	$(14 \rightarrow)$
		aka: XYZW>
341DC	5DROP	$(15 \rightarrow)$
341E8	6DROP	$(16 \rightarrow)$
341F4	7DROP	$(17 \rightarrow )$
0326E	NDROP	$(1n \#n \rightarrow )$
35FB0	#1+NDROP	(ob 1n $\#$ n $\rightarrow$ )
		aka: N+1DROP

Drops all but #n levels of the stack.  0314C DEPTH (1.n → 1.n #n) 28187 reversym (1.n #n → n1 #n) 03223 SWAP (1 2 → 2 1 1) 3576E SWAPDUP (1 2 → 2 1 1) 36885 SWAP2DUP (1 2 → 2 1 2 1) 3421A SWAPDROP (1 2 → 2 2) 35872 SWAPDROPDUP (1 2 → 2 2) 35872 SWAPDROPSWAP (1 2 3 → 3 1) aka: UNROTTROP, XYZ>ZX 341BA SWAPROT (1 2 3 → 3 2 1) aka: UNROTTSWAP, XYZ>ZYX 36C90 SWAP4ROLL (1 2 3 4 → 2 4 3 1) aka: UNROTTSWAP, XYZ>ZYX 3457F SWAPOVER (1 2 3 → 1 3 2 1) 35018 2SWAP (1 2 3 → 1 3 2 1) 35018 2SWAP (1 2 3 → 2 3 1 1) 3579C ROTDUP (1 2 3 → 2 3 1 1) 3579C ROTDUP (1 2 3 → 2 3 1 1) 35CA4 ROTZDUP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROPSWAP (1 2 3 → 2 3 1 3 1 1)	Addr.	Name	Description
0314C DEPTH (1n → 1n #n) 28187 reversym (1n #n → n1 #n) 03223 SWAP (1 2 → 2 1) 3576E SWAPDUP (1 2 → 2 1 1) 368B5 SWAPZDUP (1 2 → 2 1 2 1) 3421A SWAPDROP (1 2 → 2 2) 3421A SWAPDROPDUP (1 2 → 2 2) 35872 SWAPDROPSWAP (1 2 3 → 3 1) aka: UNROTDROP, XYZ>ZX 341BA SWAPROT (1 2 3 → 3 2 1) aka: UNROTSWAP, XYZ>ZYX 36C90 SWAP4ROLL (1 2 3 4 → 2 4 3 1) aka: XYZW>YWZX 3457F SWAPOVER (1 2 3 → 1 3 2 1) 35018 2SWAP (1 2 3 → 1 3 2 1) 35018 2SWAP (1 2 3 → 2 3 1 1) 3579C ROTDUP (1 2 3 → 2 3 1 1) 3579C ROTDUP (1 2 3 → 2 3 1 1) 352A4 ROTZDUP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 1 3 ) aka: XYZ>YZ 3574D ROTZDROP (1 2 3 → 2 1 3 ) aka: XYZ>YZ 34195 ROTTOROPSWAP (1 2 3 → 2 1 3 ) aka: XYZ>ZY 34106 ROTSWAP (1 2 3 → 2 1 3 ) aka: XYZ>ZY 34107 ROTOROPSWAP (1 2 3 → 2 1 3 ) aka: XYZ>YZ 34108 ROTROTZDROP (1 2 3 → 2 1 3 ) aka: XYZ>YZ 34108 ROTROTZDROP (1 2 3 → 2 3 1 3 1) 3423A 4ROLL (1 2 3 4 → 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX 3588B 4ROLLDROP (1 2 3 4 → 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX	2F0A1	RESETDEPTH	( ob1obn obn+1obx $\#n \rightarrow ob1obn$
0314C DEPTH (1n → 1n #n) 28187 reversym (1n #n → n1 #n) 03223 SWAP (1 2 → 2 1) 3576E SWAPDUP (1 2 → 2 1 1) 368B5 SWAPZDUP (1 2 → 2 1 2 1) 3421A SWAPDROP (1 2 → 2 2) aka: XY>Y  35857 SWAPDROPDUP (1 2 → 2 2) 35872 SWAPDROPSWAP (1 2 3 → 3 1) aka: UNROTDROP, XYZ>ZX 341BA SWAPROT (1 2 3 → 3 2 1) aka: UNROTSWAP, XYZ>ZYX 36C90 SWAP4ROLL (1 2 3 4 → 2 4 3 1) aka: XYZW>YWZX 3457F SWAPOVER (1 2 3 → 1 3 2 1) 35018 2SWAP (1 2 3 → 1 3 2 1) 35018 2SWAP (1 2 3 → 2 3 1 1) 35018 2SWAP (1 2 3 → 2 3 1 1) 3579C ROTDUP (1 2 3 → 2 3 1 1) 3579C ROTDUP (1 2 3 → 2 3 1 3 1) 352A4 ROT2DUP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 1 3 ) aka: XYZ>YZ 3574D ROT2DROP (1 2 3 → 2 1 3 ) aka: XYZ>YZ 34195 ROTTOROPSWAP (1 2 3 → 2 1 3 ) aka: XYZ>ZY 3416E ROTSWAP (1 2 3 → 2 3 1 3 ) aka: XYZ>ZY 3416B ROTROT2DROP (1 2 3 → 2 3 1 3 ) aka: XYZ>YZ 343BD ROTROT2DROP (1 2 3 → 2 3 1 3 ) aka: XYZ>YZ 343BD ROTROT2DROP (1 2 3 → 2 3 1 3 ) aka: XYZ>YZ 35CCC ROTOVER (1 2 3 → 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX 3588B 4ROLLDROP (1 2 3 4 → 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX			)
28187 reversym (1n #n → n1 #n)  03223 SWAP (1 2 → 2 1 )  3576E SWAPDUP (1 2 → 2 1 1)  36885 SWAPZDUP (1 2 → 2 1 2 1)  3421A SWAPDROP (1 2 → 2 2)  3421A SWAPDROP (1 2 → 2 2)  35857 SWAPDROPDUP (1 2 → 2 2)  35872 SWAPDROPSWAP (1 2 3 → 3 1)  aka: UNROTDROP, XYZ>ZX  341BA SWAPROT (1 2 3 → 3 2 1)  aka: UNROTSWAP, XYZ>ZYX  36C90 SWAP4ROLL (1 2 3 4 → 2 4 3 1)  aka: XYZW>YWZX  3457F SWAPOVER (1 2 → 2 1 2)  aka: DUPUNROT  36C88 SWAP3PICK (1 2 3 → 1 3 2 1)  35018 2SWAP (1 2 3 → 2 3 1)  3579C ROTDUP (1 2 3 → 2 3 1 1)  3579C ROTDUP (1 2 3 → 2 3 1 1)  35CA4 ROT2DUP (1 2 3 → 2 3 1 1)  341A8 ROTDROP (1 2 3 → 2 3 1 3 1)  341A8 ROTDROP (1 2 3 → 2 3 )  aka: XYZ>YZ  3574D ROT2DROP (1 2 3 → 2 3 )  aka: DROPSWAPDROP, XYZ>Y  34195 ROTDROPSWAP (1 2 3 → 2 1 3)  aka: DROPSWAPDROP, XYZ>Y  3416E ROTSWAP (1 2 3 → 2 1 3)  aka: XYZ>ZYZ  343BD ROTROT2DROP (1 2 3 → 3 2 1 3)  aka: YYZ>ZYZ  343BD ROTROT2DROP (1 2 3 → 2 3 1 3)  aka: YYZ>ZYZ  343BD ROTROT2DROP (1 2 3 → 2 3 1 3)  aka: YYZ>ZYZ  3416E ROTSWAP (1 2 3 → 2 1 3)  aka: YYZ>ZYZ  3423A 4ROLL (1 2 3 → 2 3 4 1)  aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1 2 3 4 → 2 3 4 1)			Drops all but #n levels of the stack.
03223 SWAP (1 2 → 2 1 ) 3576E SWAPDUP (1 2 → 2 1 1 ) 368B5 SWAPZDUP (1 2 → 2 1 2 1 ) 3421A SWAPDROP (1 2 → 2 2 ) aka: XY>Y 35857 SWAPDROPDUP (1 2 → 2 2 ) 35872 SWAPDROPSWAP (1 2 3 → 3 1 ) aka: UNROTDROP, XYZ>ZX 341BA SWAPROT (1 2 3 → 3 2 1 ) aka: UNROTSWAP, XYZ>ZYX 36C90 SWAP4ROLL (1 2 3 4 → 2 4 3 1 ) aka: XYZW>YWZX 3457F SWAPOVER (1 2 3 → 1 3 2 1 ) 35018 SWAP3PICK (1 2 3 → 1 3 2 1 ) 35018 ZSWAP (1 2 3 4 → 3 4 1 2 ) 03295 ROT (1 2 3 → 2 3 1 1 ) 3579C ROTDUP (1 2 3 → 2 3 1 1 ) 3579C ROTDUP (1 2 3 → 2 3 1 1 ) 3541A8 ROTDROP (1 2 3 → 2 3 1 3 1 ) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1 ) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1 ) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1 ) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1 ) 341BB ROTDROPSWAP (1 2 3 → 2 1 3 ) aka: XYZ>YZ 3574D ROTZDROP (1 2 3 → 2 1 3 ) aka: XYZ>YZ 3416E ROTSWAP (1 2 3 → 2 1 3 ) aka: XYZ>ZY 3416B ROTROT2DROP (1 2 3 → 2 3 1 3 ) aka: XYZ>YZ 343BD ROTROT2DROP (1 2 3 → 2 3 1 3 ) aka: XYZ>YZ 343BD ROTROT2DROP (1 2 3 → 2 3 1 3 ) aka: UNROTZDROP, XYZ>Z 35CCC ROTOVER (1 2 3 → 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX 3588B 4ROLLDROP (1 2 3 4 → 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX	0314C	DEPTH	( $1n \rightarrow 1n \ \#n$ )
3576E SWAPDUP (1 2 → 2 1 1 ) 368B5 SWAP2DUP (1 2 → 2 1 2 1 ) 3421A SWAPDROP (1 2 → 2 )	28187	reversym	( 1n $\#n \rightarrow n1 \#n$ )
368B5 SWAP2DUP (1 2 → 2 1 2 1 )  3421A SWAPDROP (1 2 → 2 )  aka: XY>Y  35857 SWAPDROPDUP (1 2 → 2 2 )  35872 SWAPDROPSWAP (1 2 3 → 3 1 )  aka: UNROTDROP, XYZ>ZX  341BA SWAPROT (1 2 3 → 3 2 1 )  aka: UNROTSWAP, XYZ>ZYX  36C90 SWAP4ROLL (1 2 3 4 → 2 4 3 1 )  aka: XYZW>YWZX  3457F SWAPOVER (1 2 → 2 1 2 )  aka: DUPUNROT  36CB8 SWAP3PICK (1 2 3 → 1 3 2 1 )  35018 2SWAP (1 2 3 → 2 3 1 )  3579C ROTDUP (1 2 3 → 2 3 1 1 )  3579C ROTDUP (1 2 3 → 2 3 1 1 )  35CA4 ROT2DUP (1 2 3 → 2 3 1 3 1 )  341A8 ROTDROP (1 2 3 → 2 3 1 3 1 )  341A8 ROTDROP (1 2 3 → 2 3 2 )  aka: XYZZ>YZ  3574D ROT2DROP (1 2 3 → 2 )  aka: DROPSWAPDROP, XYZ>Y  34195 ROTDROPSWAP (1 2 3 → 2 )  aka: XYZ>ZYZ  3416E ROTSWAP (1 2 3 → 2 1 3 )  aka: XYZ>ZYZ  343BD ROTROT2DROP (1 2 3 → 3 )  aka: XYZ>YZZ  343BD ROTROT2DROP (1 2 3 → 3 )  aka: XYZ>YZZ  355CCC ROTOVER (1 2 3 → 2 3 4 1 )  aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1 2 3 4 → 2 3 4 1 )  aka: FOURROLL, XYZW>YZWX	03223	SWAP	( 1 2 $\rightarrow$ 2 1 )
3421A SWAPDROP			( 1 2 $\rightarrow$ 2 1 1 )
3421A SWAPDROP	368B5	SWAP2DUP	$( 1 2 \rightarrow 2 1 2 1 )$
35857 SWAPDROPDUP (1 2 → 2 2) 35872 SWAPDROPSWAP (1 2 3 → 3 1) aka: UNROTDROP, XYZ>ZX 341BA SWAPROT (1 2 3 → 3 2 1) aka: UNROTSWAP, XYZ>ZYX 36C90 SWAP4ROLL (1 2 3 4 → 2 4 3 1) aka: XYZW>YWZX 3457F SWAPOVER (1 2 → 2 1 2) aka: DUPUNROT  36CB8 SWAP3PICK (1 2 3 → 1 3 2 1) 35018 ZSWAP (1 2 3 4 → 3 4 1 2) 03295 ROT (1 2 3 → 2 3 1) 3579C ROTDUP (1 2 3 → 2 3 1 1) 3579C ROTDUP (1 2 3 → 2 3 1 1) 35CA4 ROT2DUP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 ) aka: XYZ>YZ 3574D ROT2DROP (1 2 3 → 2 ) aka: DROPSWAPDROP, XYZ>Y 34195 ROTTOROPSWAP (1 2 3 → 3 2) aka: XYZ>ZY 3416E ROTSWAP (1 2 3 → 3 2) aka: XYZ>ZYZ 343BD ROTROT2DROP (1 2 3 → 3 1) aka: UNROT2DROP, XYZ>Z 35CCC ROTOVER (1 2 3 → 2 3 1 3 1) 3423A 4ROLL (1 2 3 4 → 2 3 4 1) aka: FOURROLL, XYZW>YZWX 3588B 4ROLLDROP (1 2 3 4 → 2 3 4 1)			( 1 2 $\rightarrow$ 2 )
35872 SWAPDROPSWAP (1 2 3 → 3 1) aka: UNROTDROP, XYZ>ZX  341BA SWAPROT (1 2 3 → 3 2 1) aka: UNROTSWAP, XYZ>ZYX  36C90 SWAP4ROLL (1 2 3 4 → 2 4 3 1) aka: XYZW>YWZX  3457F SWAPOVER (1 2 → 2 1 2) aka: DUPUNROT  36CB8 SWAP3PICK (1 2 3 → 1 3 2 1) 35018 2SWAP (1 2 3 4 → 3 4 1 2) 03295 ROT (1 2 3 → 2 3 1) 3579C ROTDUP (1 2 3 → 2 3 1 1) 35CA4 ROT2DUP (1 2 3 → 2 3 1 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3) aka: XYZ>YZ  3574D ROT2DROP (1 2 3 → 2 ) aka: DROPSWAPDROP, XYZ>Y  34195 ROTDROPSWAP (1 2 3 → 3 2) aka: XYZ>ZYZ  3416E ROTSWAP (1 2 3 → 2 1 3) aka: XYZ>ZYZ  343BD ROTROT2DROP (1 2 3 → 3 ) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER (1 2 3 → 2 3 1 3) 3423A 4ROLL (1 2 3 4 → 2 3 4 1) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1 2 3 4 → 2 3 4 1)			aka: XY>Y
aka: UNROTDROP, XYZ>ZX  341BA SWAPROT (1 2 3 → 3 2 1 ) aka: UNROTSWAP, XYZ>ZYX  36C90 SWAP4ROLL (1 2 3 4 → 2 4 3 1 ) aka: XYZW>YWZX  3457F SWAPOVER (1 2 → 2 1 2 ) aka: DUPUNROT  36CB8 SWAP3PICK (1 2 3 → 1 3 2 1 ) 35018 2SWAP (1 2 3 → 2 3 1 ) 3579C ROT (1 2 3 → 2 3 1 ) 3579C ROTDUP (1 2 3 → 2 3 1 1) 35CA4 ROT2DUP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 ) aka: XYZ>YZ  3574D ROT2DROP (1 2 3 → 3 2 ) aka: XYZ>ZY  34195 ROTDROPSWAP (1 2 3 → 3 2 ) aka: XYZ>ZY  3416E ROTSWAP (1 2 3 → 3 2 ) aka: XYZ>ZY  3416E ROTSWAP (1 2 3 → 3 1 3 ) aka: XYZ>ZY  343BD ROTROT2DROP (1 2 3 → 3 3 ) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER (1 2 3 → 2 3 1 3 ) 3423A 4ROLL (1 2 3 4 → 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1 2 3 4 → 2 3 4 )	35857	SWAPDROPDUP	( 1 2 $\rightarrow$ 2 2 )
aka: UNROTDROP, XYZ>ZX  341BA SWAPROT (1 2 3 → 3 2 1 ) aka: UNROTSWAP, XYZ>ZYX  36C90 SWAP4ROLL (1 2 3 4 → 2 4 3 1 ) aka: XYZW>YWZX  3457F SWAPOVER (1 2 → 2 1 2 ) aka: DUPUNROT  36CB8 SWAP3PICK (1 2 3 → 1 3 2 1 ) 35018 2SWAP (1 2 3 → 2 3 1 ) 3579C ROT (1 2 3 → 2 3 1 ) 3579C ROTDUP (1 2 3 → 2 3 1 1) 35CA4 ROT2DUP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3 ) aka: XYZ>YZ  3574D ROT2DROP (1 2 3 → 3 2 ) aka: XYZ>ZY  34195 ROTDROPSWAP (1 2 3 → 3 2 ) aka: XYZ>ZY  3416E ROTSWAP (1 2 3 → 3 2 ) aka: XYZ>ZY  3416E ROTSWAP (1 2 3 → 3 1 3 ) aka: XYZ>ZY  343BD ROTROT2DROP (1 2 3 → 3 3 ) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER (1 2 3 → 2 3 1 3 ) 3423A 4ROLL (1 2 3 4 → 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1 2 3 4 → 2 3 4 )	35872	SWAPDROPSWAP	$(123 \rightarrow 31)$
aka: UNROTSWAP, XYZ>ZYX  36C90 SWAP4ROLL (1234→2431) aka: XYZW>YWZX  3457F SWAPOVER (12→212) aka: DUPUNROT  36CB8 SWAP3PICK (123→1321) 35018 2SWAP (1234→3412) 03295 ROT (1233→231) 3579C ROTDUP (123→2311) 35CA4 ROT2DUP (123→2311) 341A8 ROTDROP (123→23131) 341A8 ROTDROP (123→23) aka: XYZ>YZ  3574D ROT2DROP (123→23) aka: XYZ>YZ  34195 ROTDROPSWAP (123→3) aka: XYZ>ZY  3416E ROTSWAP (123→3) aka: XYZ>ZY  3416E ROTSWAP (123→3) aka: XYZ>ZY  343BD ROTROT2DROP (123→3) aka: XYZ>YZ  35CCC ROTOVER (123→3) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER (123→3) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1234→234)			aka: UNROTDROP, XYZ>ZX
36C90 SWAP4ROLL (1234→2431) aka: XYZW>YWZX  3457F SWAPOVER (12→212) aka: DUPUNROT  36CB8 SWAP3PICK (123→1321) 35018 2SWAP (1234→3412) 03295 ROT (123→231) 3579C ROTDUP (123→2311) 3579C ROTDUP (123→2311) 35CA4 ROT2DUP (123→23131) 341A8 ROTDROP (123→23131) 341A8 ROTDROP (123→23) aka: XYZ>YZ  3574D ROT2DROP (123→23) aka: XYZ>YZ  34195 ROTDROPSWAP (123→32) aka: XYZ>ZY  3416E ROTSWAP (123→313) aka: XYZ>ZY  343BD ROTROT2DROP (123→31) aka: XYZ>YZZ  35CCC ROTOVER (123→31) 3423A 4ROLL (1234→2341) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1234→2341)	341BA	SWAPROT	
36C90 SWAP4ROLL (1234→2431) aka: XYZW>YWZX  3457F SWAPOVER (12→212) aka: DUPUNROT  36CB8 SWAP3PICK (123→1321) 35018 2SWAP (1234→3412) 03295 ROT (123→231) 3579C ROTDUP (123→2311) 3579C ROTDUP (123→2311) 35CA4 ROT2DUP (123→23131) 341A8 ROTDROP (123→23131) 341A8 ROTDROP (123→23) aka: XYZ>YZ  3574D ROT2DROP (123→23) aka: XYZ>YZ  34195 ROTDROPSWAP (123→32) aka: XYZ>ZY  3416E ROTSWAP (123→313) aka: XYZ>ZY  343BD ROTROT2DROP (123→31) aka: XYZ>YZZ  35CCC ROTOVER (123→31) 3423A 4ROLL (1234→2341) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1234→2341)			aka: UNROTSWAP, XYZ>ZYX
3457F SWAPOVER (12 → 212) aka: DUPUNROT  36CB8 SWAP3PICK (123 → 1321) 35018 2SWAP (123 → 231) 3579C ROT (123 → 2311) 3579C ROTDUP (123 → 2311) 35CA4 ROT2DUP (123 → 23131) 341A8 ROTDROP (123 → 23) aka: XYZ>YZ  3574D ROT2DROP (123 → 2) aka: DROPSWAPDROP, XYZ>Y  34195 ROTDROPSWAP (123 → 32) aka: XYZ>ZY  3416E ROTSWAP (123 → 213) aka: XYZ>ZY  343BD ROTROT2DROP (123 → 3) aka: XYZ>YZ  343BD ROTROT2DROP (123 → 3) aka: HOLLDROP, XYZ>Z  35CCC ROTOVER (123 → 2313) 3423A 4ROLL (123 → 2341) aka: FOURROLL, XYZW>YZWX	36C90	SWAP4ROLL	
aka: DUPUNROT  36CB8 SWAP3PICK (123→1321)  35018 2SWAP (1234→3412)  03295 ROT (123→231)  3579C ROTDUP (123→2311)  35CA4 ROT2DUP (123→23131)  341A8 ROTDROP (123→23)  aka: XYZ>YZ  3574D ROT2DROP (123→2)  aka: DROPSWAPDROP, XYZ>Y  34195 ROTDROPSWAP (123→32)  aka: XYZ>ZY  3416E ROTSWAP (123→213)  aka: XYZ>YZ  343BD ROTROT2DROP (123→313)  aka: XYZ>YXZ  343BD ROTROT2DROP (123→3)  aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER (123→213)  3423A 4ROLL (123 → 2341)  aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1234→2341)			aka: XYZW>YWZX
36CB8 SWAP3PICK (1 2 3 → 1 3 2 1) 35018 2SWAP (1 2 3 4 → 3 4 1 2) 03295 ROT (1 2 3 → 2 3 1) 3579C ROTDUP (1 2 3 → 2 3 1 1) 35CA4 ROT2DUP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3) aka: XYZ>YZ  3574D ROT2DROP (1 2 3 → 2) aka: DROPSWAPDROP, XYZ>Y  34195 ROTDROPSWAP (1 2 3 → 3 2) aka: XYZ>ZY  3416E ROTSWAP (1 2 3 → 2 1 3) aka: XYZ>ZY  343BD ROTROT2DROP (1 2 3 → 3 ) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER (1 2 3 → 2 3 1 3) 3423A 4ROLL (1 2 3 4 → 2 3 4 1) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1 2 3 4 → 2 3 4 )	3457F	SWAPOVER	$(12 \rightarrow 212)$
35018 2SWAP (1 2 3 4 → 3 4 1 2) 03295 ROT (1 2 3 → 2 3 1) 3579C ROTDUP (1 2 3 → 2 3 1 1) 35CA4 ROT2DUP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3) aka: XYZ>YZ  3574D ROT2DROP (1 2 3 → 2) aka: DROPSWAPDROP, XYZ>Y  34195 ROTDROPSWAP (1 2 3 → 3 2) aka: XYZ>ZY  3416E ROTSWAP (1 2 3 → 2 1 3) aka: XYZ>ZY  343BD ROTROT2DROP (1 2 3 → 3 ) aka: XYZ>YXZ  343BD ROTROT2DROP (1 2 3 → 3) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER (1 2 3 → 2 3 1 3) 3423A 4ROLL (1 2 3 4 → 2 3 4 1) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1 2 3 4 → 2 3 4 )			aka: DUPUNROT
35018 2SWAP (1 2 3 4 → 3 4 1 2) 03295 ROT (1 2 3 → 2 3 1) 3579C ROTDUP (1 2 3 → 2 3 1 1) 35CA4 ROT2DUP (1 2 3 → 2 3 1 3 1) 341A8 ROTDROP (1 2 3 → 2 3) aka: XYZ>YZ  3574D ROT2DROP (1 2 3 → 2) aka: DROPSWAPDROP, XYZ>Y  34195 ROTDROPSWAP (1 2 3 → 3 2) aka: XYZ>ZY  3416E ROTSWAP (1 2 3 → 2 1 3) aka: XYZ>ZY  343BD ROTROT2DROP (1 2 3 → 3 ) aka: XYZ>YXZ  343BD ROTROT2DROP (1 2 3 → 3) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER (1 2 3 → 2 3 1 3) 3423A 4ROLL (1 2 3 4 → 2 3 4 1) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1 2 3 4 → 2 3 4 )	36CB8	SWAP3PICK	$(123 \rightarrow 1321)$
3579C ROTDUP (123 → 2311) 35CA4 ROT2DUP (123 → 23131) 341A8 ROTDROP (123 → 23) aka: XYZ>YZ  3574D ROT2DROP (123 → 2) aka: DROPSWAPDROP, XYZ>Y  34195 ROTDROPSWAP (123 → 32) aka: XYZ>ZY  3416E ROTSWAP (123 → 213) aka: XYZ>YXZ  343BD ROTROT2DROP (123 → 3) aka: XYZ>YXZ  343BD ROTROT2DROP (123 → 3) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER (123 → 2313) 3423A 4ROLL (1234 → 2341) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1234 → 234)	35018	2SWAP	$( 1 2 3 4 \rightarrow 3 4 1 2 )$
35CA4 ROT2DUP (123→23131) 341A8 ROTDROP (123→23) aka: XYZ>YZ  3574D ROT2DROP (123→2) aka: DROPSWAPDROP, XYZ>Y  34195 ROTDROPSWAP (123→32) aka: XYZ>ZY  3416E ROTSWAP (123→213) aka: XYZ>YXZ  343BD ROTROT2DROP (123→3) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER (123→2313) 3423A 4ROLL (123 → 2313) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP (1234→234)	03295		
341A8 ROTDROP (1 2 3 $\rightarrow$ 2 3 )	3579C	ROTDUP	$( 1 2 3 \rightarrow 2 3 1 1 )$
341A8 ROTDROP (1 2 3 $\rightarrow$ 2 3 )	35CA4	ROT2DUP	$( 1 2 3 \rightarrow 2 3 1 3 1 )$
3574D ROT2DROP ( 1 2 3 $\rightarrow$ 2 )			$(123 \rightarrow 23)$
aka: DROPSWAPDROP, XYZ>Y         34195       ROTDROPSWAP $(123 \rightarrow 32)$ aka: XYZ>ZY         3416E       ROTSWAP $(123 \rightarrow 213)$ aka: XYZ>YXZ         343BD       ROTROT2DROP $(123 \rightarrow 3)$ aka: UNROT2DROP, XYZ>Z         35CCC       ROTOVER $(123 \rightarrow 2313)$ 3423A       4ROLL $(1234 \rightarrow 2341)$ aka: FOURROLL, XYZW>YZWX         3588B       4ROLLDROP $(1234 \rightarrow 234)$			aka: XYZ>YZ
34195 ROTDROPSWAP ( 1 2 3 $\rightarrow$ 3 2 ) aka: XYZ>ZY  3416E ROTSWAP ( 1 2 3 $\rightarrow$ 2 1 3 ) aka: XYZ>YXZ  343BD ROTROT2DROP ( 1 2 3 $\rightarrow$ 3 ) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER ( 1 2 3 $\rightarrow$ 2 3 1 3 ) 3423A 4ROLL ( 1 2 3 4 $\rightarrow$ 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP ( 1 2 3 4 $\rightarrow$ 2 3 4 )	3574D	ROT2DROP	$(123 \rightarrow 2)$
34195 ROTDROPSWAP ( 1 2 3 $\rightarrow$ 3 2 ) aka: XYZ>ZY  3416E ROTSWAP ( 1 2 3 $\rightarrow$ 2 1 3 ) aka: XYZ>YXZ  343BD ROTROT2DROP ( 1 2 3 $\rightarrow$ 3 ) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER ( 1 2 3 $\rightarrow$ 2 3 1 3 ) 3423A 4ROLL ( 1 2 3 4 $\rightarrow$ 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP ( 1 2 3 4 $\rightarrow$ 2 3 4 )			aka: DROPSWAPDROP, XYZ>Y
3416E ROTSWAP ( 1 2 3 $\rightarrow$ 2 1 3 ) aka: XYZ>YXZ  343BD ROTROT2DROP ( 1 2 3 $\rightarrow$ 3 ) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER ( 1 2 3 $\rightarrow$ 2 3 1 3 ) 3423A 4ROLL ( 1 2 3 4 $\rightarrow$ 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP ( 1 2 3 4 $\rightarrow$ 2 3 4 )	34195	ROTDROPSWAP	
aka: XYZ>YXZ         343BD       ROTROT2DROP $(123 \rightarrow 3)$ aka: UNROT2DROP, XYZ>Z         35CCC       ROTOVER $(123 \rightarrow 2313)$ 3423A       4ROLL $(1234 \rightarrow 2341)$ aka: FOURROLL, XYZW>YZWX         3588B       4ROLLDROP $(1234 \rightarrow 234)$			aka: XYZ>ZY
343BD ROTROT2DROP ( 1 2 3 $\rightarrow$ 3 ) aka: UNROT2DROP, XYZ>Z  35CCC ROTOVER ( 1 2 3 $\rightarrow$ 2 3 1 3 ) 3423A 4ROLL ( 1 2 3 4 $\rightarrow$ 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX  3588B 4ROLLDROP ( 1 2 3 4 $\rightarrow$ 2 3 4 )	3416E	ROTSWAP	$(123 \rightarrow 213)$
aka: UNROT2DROP, XYZ>Z         35CCC       ROTOVER       ( 1 2 3 $\rightarrow$ 2 3 1 3 )         3423A       4ROLL       ( 1 2 3 4 $\rightarrow$ 2 3 4 1 )         aka: FOURROLL, XYZW>YZWX         3588B       4ROLLDROP       ( 1 2 3 4 $\rightarrow$ 2 3 4 )			aka: XYZ>YXZ
aka: UNROT2DROP, XYZ>Z         35CCC       ROTOVER       ( 1 2 3 $\rightarrow$ 2 3 1 3 )         3423A       4ROLL       ( 1 2 3 4 $\rightarrow$ 2 3 4 1 )         aka: FOURROLL, XYZW>YZWX         3588B       4ROLLDROP       ( 1 2 3 4 $\rightarrow$ 2 3 4 )	343BD	ROTROT2DROP	$(123 \rightarrow 3)$
35CCC ROTOVER ( 1 2 3 $\rightarrow$ 2 3 1 3 ) 3423A 4ROLL ( 1 2 3 4 $\rightarrow$ 2 3 4 1 ) aka: FOURROLL, XYZW>YZWX 3588B 4ROLLDROP ( 1 2 3 4 $\rightarrow$ 2 3 4 )			
3423A 4ROLL ( 1 2 3 4 $\rightarrow$ 2 3 4 1 ) aka: FOURROLL, XYZW $>$ YZWX 3588B 4ROLLDROP ( 1 2 3 4 $\rightarrow$ 2 3 4 )	35CCC	ROTOVER	·
aka: FOURROLL, XYZW>YZWX 3588B 4ROLLDROP ( 1 2 3 4 $ ightarrow$ 2 3 4 )			
3588B 4ROLLDROP ( 1 2 3 4 $\rightarrow$ 2 3 4 )			· · · · · · · · · · · · · · · · · · ·
	3588B	4ROLLDROP	
	35F06	4ROLLSWAP	$(1234 \rightarrow 2314)$

Addr.	Name	Description
36043	4ROLLROT	$( 1 2 3 4 \rightarrow 2 4 1 3 )$
		aka: FOURROLLROT
360E3	4ROLLOVER	$( 1 2 3 4 \rightarrow 2 3 4 1 4 )$
34257	5ROLL	$(\ 1\ 2\ 3\ 4\ 5\ \rightarrow\ 2\ 3\ 4\ 5\ 1\ )$
		aka: FIVEROLL
358A7	5ROLLDROP	$( \ 1 \ 2 \ 3 \ 4 \ 5 \  ightarrow \ 2 \ 3 \ 4 \ 5 \ )$
34281	6ROLL	$(16 \rightarrow 261)$
		aka: SIXROLL
342EA	7ROLL	$(17 \rightarrow 27 1)$
		aka: SEVENROLL
342BB	8ROLL	$(18 \rightarrow 281)$
		aka: EIGHTROLL
03325	ROLL	$(1n \#n \rightarrow 2n 1)$
35FC4	ROLLDROP	$(1n \#n \rightarrow 2n)$
35D80	ROLLSWAP	$(1n \#n \rightarrow 2n-1 1 n)$
344F2	#1+ROLL	( ob 1n $\#n \rightarrow 1n$ ob )
34517	#2+ROLL	(ab1n $\#n \rightarrow b1n a$ )
2D6006	^#3+ROLL	( obn+3obnob1 #n $\rightarrow$
		obn+2ob1 obn+3 )
344DD	#+ROLL	( 1n+m #n #m $\rightarrow$ 2n+m 1 )
344CB		( 1n-m #n #m $\rightarrow$ 2n-m 1 )
3422B	UNROT	$(123 \rightarrow 312)$
		aka: 3UNROLL, XYZ>ZXY
	UNROTDUP	$(123 \rightarrow 3121)$
35872	UNROTDROP	$(123 \to 31)$
		aka: SWAPDROPSWAP, XYZ>ZX
343BD	UNROT2DROP	$(\begin{array}{cccccccccccccccccccccccccccccccccccc$
		aka: ROTROT2DROP, XYZ>Z
341BA	UNROTSWAP	$(123 \rightarrow 321)$
		aka: SWAPROT, XYZ>ZYX
	UNROTOVER	$(123 \rightarrow 3121)$
3422B	3UNROLL	$(123 \to 312)$
	•	aka: UNROT, XYZ>ZXY
34331	4UNROLL	$(1234 \rightarrow 4123)$
25544	4	aka: FOURUNROLL, XYZW>WXYZ
	4UNROLLDUP	
343CF	4UNROLL3DROP	· ·
26057	ALIMID OL T D OFF	aka: XYZW>W
3605/	4UNROLLROT	$( 1 2 3 4 \rightarrow 4 3 2 1 )$

```
Addr.
         Name
                          Description
34357
         5UNROLL
                          (12345 \rightarrow 51234)
                          aka: FIVEUNROLL
3438D
                          (1..6 \rightarrow 61..5)
         6UNROLL
                          aka: SIXUNROLL
35BEB
         7UNROLL
                          (1..7 \rightarrow 7 1..6)
3615B
         8UNROLL
                          (1..8 \rightarrow 8 1..7)
                          (1..9 \rightarrow 9 1..8)
28225
         (9UNROLL)
                          (1..10 \rightarrow 10 1..9)
3616F
         10UNROLL
                          (1..n \#n \rightarrow n 1..n-1)
0339E
         UNROLL
                          ( ob 1..n \#n \to n ob 1..n-1 )
34552
         #1+UNROLL
34564
         #2+UNROLL
                          (ab1..n \#n \rightarrow nab1..n-1)
                          (1..n+m #n #m \rightarrow n+m 1..n+m-1)
3453D
         #+UNROLL
3452B
         #-UNROLL
                          (1..n-m \# n \# m \rightarrow n-m 1..n+m-1)
032C2
         OVER
                          (12 \rightarrow 121)
35CF4
                          (12 \rightarrow 1211)
         OVERDUP
                          (12 \rightarrow 112)
35D6C
         OVERSWAP
                          aka: OVERUNROT
                          (12 \rightarrow 112)
35D6C
         OVERUNROT
                          aka: OVERSWAP
36CF4
                          (1234 \rightarrow 123431)
         OVER5PICK
                          (1234 \rightarrow 123412)
37046
         20VER
                          (123 \rightarrow 1231)
34485
         3PICK
                          (123 \rightarrow 1213)
35F1A
         3PICKSWAP
360F7
         3PICKOVER
                          (123 \rightarrow 12313)
                          (123 \rightarrow 12312)
36CCC
         3PICK3PICK
                          (1234 \rightarrow 1231)
2F1C6
         DROP3PICK
                          (1234 \rightarrow 12341)
3448A
         4PICK
                          (1234 \rightarrow 12314)
35F2E
         4PICKSWAP
36CE0
         SWAP4PICK
                          (1234 \rightarrow 12431)
                          (1 2 3 4 \rightarrow 1 2)
                                              3 4 1 4 )
3610B
         4PICKOVER
                          (\ 1\ 2\ 3\ 4\ 5\ \rightarrow\ 1\ 2\ 3\ 4\ 5\ 1\ )
3448F
         5PICK
34494
         6PICK
                          (1..6 \rightarrow 1..6 1)
                          (1..7 \rightarrow 1..7 1)
34499
         7PICK
                          (1..8 \rightarrow 1..8 1)
3449E
         8PICK
344A3
                          (1..9 \rightarrow 1..9 1)
         (9PICK)
                          (1..10 \rightarrow 1..10 1)
344A8
         (10PICK)
032E2
         PICK
                          (1..n #n \rightarrow 1..n 1)
                          (1..n \#n-1 \rightarrow 1..n 1)
34436
         #1+PICK
                          (1..n \#n-2 \rightarrow 1..n 1)
34451
         #2+PICK
```

Addr.	Name	Description
34465	#3+PICK	$(1n \#n-3 \rightarrow 1n 1)$
34474	#4+PICK	( 1n $\#n-4 \rightarrow 1n 1$ )
34417	#+PICK	( 1n+m #n #m $ ightarrow$ 1n+m 1 )
34405	#-PICK	( $1n-m$ $\#n$ $\#m \rightarrow 1n-m$ 1 )

# **Chapter 18**

# **Temporary Environments**

System RPL local variables (also known as temporary or lambda variables) work in the same way and have the same uses as in User RPL. You assign values to them, and these values can be recalled or changed while the variables exist. The stored values are referenced by means of local identifiers (also called lambda identifiers, or lams for short). These are very similar to the global identifiers that reference variables stored in memory (see Chapter 8), but the variables exist only temporarily.

But there is one difference: in System RPL you can give a null (that is, empty) name to local variables, therefore effectively making them unnamed variables. This saves memory and is much faster. But before learning how to create and use unnamed local variables, let us learn how to use normal, named ones.

## 18.1 Named Local Variables

Creating named local variables is very similar to creating temporary variables in User RPL. You have to create a list of local identifiers (called lams for short), and run the command BIND. To recall the contents of one of them, just enter its local identifier. To store a new value, put that value and the lam in the stack, and run STO. To remove the local variables from memory, use ABND (shortcut for "abandon"). The code is not checked for matching BIND/ABND, so you may include them in different programs if you wish. But this also means you must be sure to have an ABND for each BIND.

Here is a little program that creates two local variables, recalls their contents and assigns new values for them (it is called LAM1):

```
5
        LAM sec
                   (first contains 2, and sec 3)
      BIND
                   (recall contents from first - 2)
      LAM first
                   (recall contents from sec - 3)
      LAM sec
10
      DUP
      ' LAM first
                   (store new contents in first)
      STO
                   (results 5)
      응+
      ' LAM sec
15
      STO
                   (store sum in sec)
      ABND
                   (delete variables from memory)
```

#### 18.2 Unnamed Local Variables

As said above, you can use unnamed local variables. Technically, they have a name: the null, or empty, name; but all "unnamed" variables have the same name. Since they cannot be identifed by name, positional syntax is necessary. The above program could be rewritten using null named temporary variables this way (now called LAM2):

```
1
    ::
       82 83
       { NULLLAM NULLLAM }
      BIND
5
                    (recalls 2)
       2GETLAM
       1GETLAM
                    (recalls 3)
      DUP
       2PUTLAM
       응+
10
      1PUTLAM
      ABND
```

The numbering is done in the same order as the stack levels: 1GETLAM contains what was on level one, 2GETLAM contains what was on level two, etc. There are supported entries to recall and store directly up to the  $22^{nd}$  variable (1GETLAM to 22GETLAM, and their PUTLAM equivalents). To access variables with numbers higher than 23 (which probably will not happen very often), use GETLAM, which takes a bint representing the variable number and returns its

contents; and PUTLAM, which takes an object and the variable number, and stores that object in the specified variable.

## 18.3 Nested Temporary Environments

It is perfectly possible to use two or more temporary environments at the same time. Nothing special needs to be done during the creation: just use another DOBIND or BIND before abandoning the previous one. When an ABND is found, it always refers to the most recent BIND.

If you only use named lams, nothing special needs to be done. There will be no confusion with names, unless you redefine an existing variable (but doing this will only make a great mess out of your program). However, when at least one of the temporary environments has unnamed lams, you must pay attention to the numbering.

Note that the GETLAM words do not necessarily refer to unnamed local variables: 1GETLAM recalls the most recently bound variable, 2GETLAM the one before that, and so on. (When binding lams, the binding starts at the stack level with the largest number, working towards the one with the smallest number, so that the last bound variable is the one whose contents where in level one.) You may use the GETLAM words also to access named lams.

Due to the nature of temporary environments, there appears to be an extra local variable (before all the others) for internal housekeeping purposes. To access the unnamed lams of a previous environment, you must add the number of variables bound in the current environment *plus one* to the number you would have used before the second binding.

The following program (named LAM3) will try to make the above explanation clearer:

```
응4
       %3
15
         NULLLAM
         NULLLAM
       BIND
       1GETLAM
                  (Returns 3)
20
       2GETLAM
                  (Returns 4)
       4GETLAM
                  (Returns 1)
       5GETLAM
                  (Returns 2)
       ABND
       ABND
25
```

First, this program binds 2 to n2 and 1 n1, but these names are never used. Instead, 1GETLAM is used to acces the most recently bound value, that is, 1, which could also be accessed via LAM n1. Following, 2GETLAM returns the next-to-last value, or 2.

Things become more complicated when another environment is bound, this time to unnamed lams. Now 1GETLAM returns 3, which belongs to the new environment, and was the last bound variable. Similarly, 2GETLAM also returns a variable bound in this second batch.

If we wanted to access the variable that previously was number one, we need to add the number of variables bound in the new environment (that is, two) plus one (the housekeeping pseudo-variable) to the previous number. So, to get what 1GETLAM would have returned before, we add three to one, obtaining 4GETLAM. And this returns, as expected, 1. Similarly, 5GETLAM returns 2, the same 2GETLAM had returned before the second binding.

Naturally, after the first ABND (corresponding to the binding of values 4 and 3), 1GETLAM and 2GETLAM would again return 1 and 2, respectively.

If you have been able to understand the above, you will not have problems to nest temporary environments when necessary.

## 18.4 Other Ways of Binding

First, instead of a list of lams, you can always put each lam in the stack, followed by the number of variables to be bound, and run the command DOBIND instead of BIND. As a matter of fact, BIND is just :: INNERCOMP DOBIND :.

When you are binding a great number of local variables, instead of entering the following code (which takes 67.5 bytes)

```
1 ... { NULLLAM } BIND ...
```

use this, which takes only 12.5 bytes, a savings of 55 bytes:

```
... NULLLAM TWENTYFOUR NDUPN {}N BIND ...
```

However, why create a composite if it is going to be exploded later? Replace  $\{\}$ N BIND for DOBIND, and save 2.5 more bytes.

Or you can also use TWENTYFOUR ' NULLLAM CACHE. However, if you use this, an extra variable is created to hold the count, so you must add one to the variable positions of the previous examples.

When decompiling code, you can sometimes find things like

```
... ZEROZEROZERO BINT3 DOBIND ...
```

which is yet another way of binding three null named variables. This works because instead of NULLLAM, any fixed address ROM object can be used, as ZERO in this example.

The following constructs are the most compact ways to create temporary environments for N null named variables.

#### N Commands to create N null named variables

- 1 1LAMBIND
- 2 ZEROZEROTWO DOBIND
- 2 FPTR2 ^2LAMBIND
- 3 FPTR2 ^3LAMBIND
- 4 4NULLLAM{} BIND
- N NULLLAM #N NDUPN DOBIND

## 18.5 Reference

#### 18.5.1 Builtin IDs and LAMs

Addr.	Name	Description
272FE	NULLID	$(  o  ext{id} )$
		Null (empty) identifier
2B3AB	NULLLAM	( $ ightarrow$ lam )
		Puts NULLLAM in the stack.
27155	'IDX	( $ ightarrow$ id )
		Puts ID X unevaluated on the stack.
272F3	(ID_EQ)	ID EQ
27937	(ID_SIGMADAT)	$\mathrm{ID}\ \Sigma \mathrm{DAT}$

#### 18.5.2 Conversion

Addr.	Name	Description
05B15	\$>ID	( $\$ \rightarrow \mathtt{ID}$ )
362DE	DUP\$>ID	( $\$ \rightarrow \$$ ID )

## **18.5.3 Temporary Environments Words**

Addr.	Name	Description
074D0	BIND	( obnob1 {lamnlam1} $\rightarrow$ )
		Binds n objects to n differently named lams.
074E4	DOBIND	( obnob1 lamnlam1 $\#$ n $ o$ )
		Binds n objects to n differently named lams.
36518	1LAMBIND	( ob $ ightarrow$ )
		Binds one object to a null named lam.
36513	DUP1LAMBIND	( ob $ ightarrow$ ob )
		Does DUP then 1LAMBIND.
155006	^2LAMBIND	( ob1 ob2 $ ightarrow$ )
		Binds two objects to null named lams.
156006	^3LAMBIND	( ob1 ob2 ob3 $ ightarrow$ )
		Binds three objects to null named lams.

Addr.	Name	Description
0DE0B0	~nNullBind	( obnob1 $\#$ n $ o$ )
		Binds #n objects to null named lams. 1LAM
		has the count, 2LAM the first object. Decom-
		piles to
		:: ' NULLLAM CACHE ;
36A77	dvarlsBIND	$( ob \rightarrow )$
		Binds ob to LAM 'dvar.
07497	ABND	$(\rightarrow)$
		Abandons topmost temporary environment.
34D00	CACHE	( obnob1 #n lam $\rightarrow$ )
		Binds all objects under the same name. 1LAM
		has the count.
34EBE	DUMP	( NULLLAM $\rightarrow$ oblobn #n )
		Inverse of CACHE. Always does garbage collec-
		tion.
34D58	SAVESTACK	$(\rightarrow)$
		Caches stack to SAVELAM.
34FA6	undo	$( \rightarrow )$
		Dumps SAVELAM.
07943	@LAM	$(lam \rightarrow ob T)$
		$($ lam $\rightarrow$ F $)$
		Tries recalling object from lam. If success-
		ful, returns object and TRUE, otherwise returns
		just FALSE.
07D1B	STOLAM	$( ob lam \rightarrow )$
		Tries storing object in lam. Generates "Unde-
		fined Local Name" error if lam is not found.
075A5	GETLAM	$( #n \rightarrow ob )$
		Gets contents of nth topmost lam.
34616	1GETLAM	$(\rightarrow \text{ ob })$
34620	2GETLAM	$(\rightarrow \text{ ob })$
3462A	3GETLAM	$(\rightarrow \text{ ob })$
34634	4GETLAM	$(\rightarrow \text{ob})$
3463E	5GETLAM	$(\rightarrow \text{ ob })$
34648	6GETLAM	$(\rightarrow \text{ ob })$
34652	7GETLAM	$(\rightarrow \text{ob})$
3465C	8GETLAM	$(\rightarrow \text{ob})$
34666	9GETLAM	$(\rightarrow \text{ob})$
34670	10GETLAM	$( \rightarrow ob )$

Addr.	Name	Description
3467A	11GETLAM	$( \rightarrow ob )$
34684	12GETLAM	$( \rightarrow ob )$
3468E	13GETLAM	$( \rightarrow ob )$
34698	14GETLAM	$( \rightarrow ob )$
346A2	15GETLAM	$( \rightarrow ob )$
346AC	16GETLAM	$( \rightarrow ob )$
346B6	17GETLAM	$( \rightarrow ob )$
346C0	18GETLAM	$( \rightarrow ob )$
346CA	19GETLAM	( $ ightarrow$ ob )
346D4	20GETLAM	( $ ightarrow$ ob )
346DE	21GETLAM	( $ ightarrow$ ob )
346E8	22GETLAM	( $ ightarrow$ ob )
346F2	(23GETLAM)	( $ ightarrow$ ob )
346FC	(24GETLAM)	( $ ightarrow$ ob )
34706	(25GETLAM)	( $ ightarrow$ ob )
34710	(26GETLAM)	( $ ightarrow$ ob )
3471A	(27GETLAM)	( $ ightarrow$ ob )
075E9	PUTLAM	( ob $\#n \rightarrow$ )
		Stores new contents to nth topmost lam.
34611	1PUTLAM	( ob $ ightarrow$ )
3461B	2PUTLAM	( ob $ ightarrow$ )
34625	3PUTLAM	( ob $ ightarrow$ )
3462F	4PUTLAM	( ob $ ightarrow$ )
34639	5PUTLAM	( ob $ ightarrow$ )
34643	6PUTLAM	( ob $ ightarrow$ )
3464D	7PUTLAM	( ob $ ightarrow$ )
34657	8PUTLAM	( ob $ ightarrow$ )
34661	9PUTLAM	( ob $ ightarrow$ )
3466B	10PUTLAM	( ob $ ightarrow$ )
34675	11PUTLAM	( ob $ ightarrow$ )
3467F	12PUTLAM	( ob $ ightarrow$ )
34689	13PUTLAM	( ob $ ightarrow$ )
34693	14PUTLAM	( ob $ ightarrow$ )
3469D	15PUTLAM	( ob $ ightarrow$ )
346A7	16PUTLAM	( ob $ ightarrow$ )
346B1	17PUTLAM	( ob $ ightarrow$ )
346BB	18PUTLAM	( ob $ ightarrow$ )
346C5	19PUTLAM	( ob $ ightarrow$ )
346CF	20PUTLAM	( ob $\rightarrow$ )

Addr.	Name	Description
346D9	21PUTLAM	( ob $\rightarrow$ )
346E3	22PUTLAM	$( ob \rightarrow )$
346ED	(23PUTLAM)	$( ob \rightarrow )$
346F7	(24PUTLAM)	( ob $\rightarrow$ )
34701	(25PUTLAM)	( ob $\rightarrow$ )
3470B	(26PUTLAM)	( ob $\rightarrow$ )
34715	(27PUTLAM)	$( ob \rightarrow )$
34797	DUP4PUTLAM	$( ob \rightarrow ob )$
		Does DUP then 4PUTLAM.
364FF	1GETABND	( $ ightarrow$ 1lamob )
		Does 1GETLAM then ABND.
35DEE	1ABNDSWAP	( ob $ ightarrow$ 1lamob ob )
		Does 1GETABND then SWAP.
35F42	1GETSWAP	( ob $ ightarrow$ 1lamob ob )
		Does 1GETLAM then SWAP.
2F318	1GETLAMSWP1+	( $\# \rightarrow 1$ lamob $\#+1$ )
		Does 1GETLAM then SWAP#1+.
3632E	2GETEVAL	$( \rightarrow ? )$
		Does 2GETLAM then EVAL.
3483E	GETLAMPAIR	( $\#n \rightarrow \#n$ ob lam F )
		( $\#n \rightarrow \#n T$ )
		Gets lam contents and name (10 = 1lam, 20 =
		2lam, etc.)
347AB	DUPTEMPENV	$( \rightarrow )$
		Duplicates topmost tempenv (clears protection
		word).
2B3A6	<pre>1NULLLAM{ }</pre>	$( \rightarrow \{\} )$
		Puts a list with one NULLLAM in the stack.
271F4	$(2NULLLAM{})$	$( \rightarrow \{\} )$
		Puts a list with two times NULLLAM in the
		stack.
27208	(3NULLLAM{})	$( \rightarrow \{\} )$
		Puts a list with three times NULLLAM in the
		stack.
2B3B7	4NULLLAM{}	$( \rightarrow \{\} )$
		Puts a list with four times NULLLAM in the
		stack.

# **Chapter 19 Runstream Control**

So far, you have only seen commands that do not affect the normal program flow. All the programs presented work sequentially, from the first command to the last, without any kind of change in this order. However, on all but the simplest programs, some kind of disruption in the default order is necessary. Sometimes, you need to have some part of the program repeated several times, or some actions must be executed only under certain conditions.

This chapter will describe some low-level entries that affect the normal execution order. The situations described above can be done with higer-level constructs such as loops (see Chapter 21) and conditionals (described in Chapter 20). And you will probally use those constructs more often than most of the entries below. However, this chapter also describes some concepts that help understanding how a System RPL program, and how to change the normal program flow.

## 19.1 Some Concepts

As you know from the Introduction, a compiled System RPL program consists of a series of pointers to address in the memory. Basically, a program is executed by jumping to the pointed address, executing whatever is there, returning back to the program, jumping to the next address, and so on.

Actually, it is more complicated, because there are also objects such as real numbers, strings and even other programs (secondaries) embedded inside the programs. This requires some "magic" (actually, just carefully written code) to be properly handled, but it is outside the scope of this document to describe how this is dealt with. Just assume that when an object is found, it is "executed". For most objects (such as real numbers or strings), this means putting themselves in the stack, for secondaries this means executing their contents, and for others such as identifiers this means trying to recall the contents and executing them, or simply putting themselves in the stack.

Since the objects are executed in order, it becomes necessary to have some kind of variable that will always point to the next object to be executed. This is called the *interpreter pointer*, and is stored in a CPU register. After each object is executed, this pointer is advanced to point to the next object.

When a DUP is found in the program, what happens is as follows: actually, the only thing that is stored is the address #03188h. A jump is made to that address. In that address, there is some piece of machine-language code. This code is executed and in the end the interpreter pointer is advanced, and a jump is made to the next object, whatever it is.

Things get slightly more complicated when one wants to execute, for example, INCOMPDROP. At this command's address, there is a secondary object, whose contents happen to be :: INNERCOMP DROP ;. The problem is that it is necessary to switch to that (sub-)program, execute all its contents, and then return back to the program in which INCOMPDROP was called. Since it is perfectly possible for a sub-program to have even more sub-programs inside it, it turns out that some kind of stack is necessary. When a secondary (or any other composite) is executed, the address of the object after this composite in the calling program is pushed into this stack. The composite is then executed, by means of the interpreter pointer pointing to each of its objects. When it finishes, an address is popped from the return stack, and execution returns there. This was the address of the next object in the previous program, so execution resumes properly. This stack is called the *return stack*.

The description above is rather incomplete, but it should give you an idea of how things work. There are many details that would make a detailed explanation of System RPL programs too long and complicated, so this detailed explanation will not be given in this book.

Another important concept is that of the *runstream*. It is the sequence of objects that follow the object currently being executed. For example, during the execution of the 'command in this program

```
:: ' DUP :: EVAL ; % 1. ;
```

the runstream contains three objects. The first is the command DUP. The second is the secondary that contains the EVAL command inside (but *not* the command EVAL or just the ::), and the third is the real number one. Several words (including ', as you will see below), take their argument from the next object in the runstream, and not from the data stack, as most commands do. So, the "argument" to ' is the command DUP.

You should now have understood why this chapter is called "Runstream Control": the commands here affect the runstream, that is, they affect the order in which the objects that form the program will be executed.

#### 19.2 **Runstream Commands**

The commands described here are the basic actions available. In the reference section below you will find several commands that combine these commands with others.

#### Command **Stack and Description**

 $(\rightarrow \text{ ob })$ 

This is very easy to understand: it pushes the object after it (that is, the first object in the runstream) in the stack. This pushed object will not be executed; execution resumes in the object after it. As an example,

```
:: %1 %2 ' %+ EVAL ;
is equivalent to
:: %1 %2 %+ ;
```

This action of pushing the next object in the stack instead of evaluating it is called *quoting* the next object.

 $(\rightarrow \text{ ob })$ 'R

> This pushes into the data stack the object that is pointed to by the topmost pointer in the return stack, and skips this pushed object. In other words, the first object in the composite that contains the composite currently being executed is pushed in the data stack, and skipped. If, however, the object that would be pushed is SEMI, then a null composite is pushed instead. As an example, the RESOROMP command is just like ROMPTR@, but its argument comes after it in the runstream (see Chapter 16). Here is how RESOROMP is defined:

```
:: 'R ROMPTR@ DROP ;
```

It just pushes the object after RESOROMP in the stack and calls ROMPTR@.

 $(\rightarrow \text{ ob TRUE})$ ticR (  $\rightarrow$  FALSE )

> This is similar to 'R, but it will not push a null composite if there was no object to be pushed; instead it returns FALSE. If an object could be pushed, it is pushed along with TRUE.

#### Command Stack and Description

>R  $(comp \rightarrow)$ 

This pushes a pointer to the body of the composite given as argument in the return stack. That means that when the current secondary ends, execution will not go back to the one that called the current composite. Before that, the composite given as argument will be executed, and only after it finishes will the execution resume at the secondary that called the current one. As an example, the code below returns in the stack the reals 3, 2 and 1, in this order:

:: ' :: % 1 ; >R % 3 % 2 ;  $(\rightarrow$  :: )

Pushes in the data stack a secondary whose contents are what is pointed to by the first pointer in the return stack, which is popped. In other words, it pushes as a secondary the rest of the commands in the secondary that called the current one. This commands will then not be executed after the current secondary finishes. As an example, the code below pushes the reals 3, 2 and 1 in the stack, in this order:

:: :: R> EVAL % 1 ; % 3 % 2 ;  $(\rightarrow$  :: )

This is the same as R>, but it does not pop the return stack. The same example of the above command, with R> changed into R@ would return 3, 2, 1, 3 and 2.

 $\texttt{IDUP} \qquad \qquad (\,\rightarrow\,)$ 

R>

R@

Pushes the interpreter pointer into the return stack. This means that after the current secondary finishes, a jump will be made to the object just after the IDUP, thereby executing the rest of the current secondary once more.

RDROP  $(\rightarrow)$ 

Pops the return stack. That is, the remaining objects in the secondary that called the current one will not be executed.

RDUP  $(\rightarrow)$ 

Duplicates the top address in the return stack.

RSWAP  $(\rightarrow)$ 

Swaps the top two addresses in the return stack.

?SEMI (flag  $\rightarrow$ )

If the flag is TRUE, skips the rest of the current secondary.

Command	Stack and Description
COLA	$(\rightarrow)$
	This executes only the next object in the runstream, skipping
	the rest of the current secondary. The program below pushes
	only 1 in the stack:
	:: COLA % 1 % 2 % 3 ;
	See below for some good uses for COLA.
SKIP	(  ightarrow )
	Skips the next object in the runstream. The program above,
	with COLA replaced by SKIP would push 2 and 3 in the stack.
?SKIP	$(\texttt{flag}  \rightarrow)$
	Does SKIP if the flag is true.

## 19.3 Some Examples

Our first example will show a useful use of COLA: when recursion is used. Suppose we have the two programs below for calculating the factorial of a number:

fact:

```
1
  ::
     CKREAL
     { LAM x } BIND
                       (First value for factorial)
     %1
5
     factiter
     ABND
 factiter:
1
  ::
     LAM x %0 = ?SEMI (Exits if x = 0)
                       (Multiplies by current value)
     LAM x %1- 'LAM x STO
     COLA factiter
5
```

Note the word COLA before the recursive invocation of factiter. Without it, the program would require many return stack levels, all of which would point to SEMI. With COLA, nothing is pushed in the return stack. factiter is simply called, without storing the address of where the interpreter should

jump back to. This makes the program always use a fixed number of return stack levels.

However, COLA is not used only in this case. It is a very useful command in other situations. Let us say that in your project you will frequently need to perform a case (see section 20.3) comparing if a real number is equal to 3. It is convenient to write a program to do this (like the built-in word %1=case) instead of repeating "%3 %= case" all the time.

A first attempt would be this program:

```
:: %3 %= case ;
```

However, this would not work. This is because case takes its arguments from the runstream, that is, the currently executed program, and not from the calling composite. This means the argument for case is ;, which is not what is desired. But there is a solution: use COLA before the case. This will drop the rest of the runstream after the command after it, in a way merging the current command with the composite that called it. So, if we add COLA before case, and embed this new sub-program in another, like this:

```
:: ... :: %3 %= COLA case ; <act_T> <act_F> ...
it is as if the code were like this:
:: ... %3 %= case <act_T> <act_F> ...
```

which is what we want. Therefore, the correct way to define our sub-program is with COLA before case. This is a frequent combination, so there is a shortcut command, COLAcase, that is equivalent to COLA followed by case. There are other words like this, see the reference below.

The next example (which uses an error-trapping structure that will be described in Chapter 22) is the command  $\ni$  from the OT49 library (see section A.3.1), written by Wolfgang Rautenberg. This command is used like this: «  $\ni$  . . . »

That is, generally as the first command in a program (which, naturally, can be a System RPL program, not only a User one). It causes the program to be executed with the display off (which makes it slightly faster). All the follows the  $\mathbb D$  until the end of the secondary is executed "blindly". When  $\mathbb D$  is run, it turns off the display, and when the secondary finishes executing, the display is turned back on. But how can this be done, if nothing special needs to be called after the program finishes? The answer is simple: by manipulating the return stack. Here is the disassembly of that command:

```
1 ::
Code
R>
```

```
ERRSET
COMPEVAL
ERRTRAP
::
'REVAL
ERRJMP
;
Code
```

The first code object turns off the display. It is a short and simple piece of machine language, but it is outside the scope of this book to describe it. Then, R> brings the rest of the composite that called D into the data stack. It is evaluated by COMPEVAL. The only difficulty in the program is that we must turn the display back on even if there was an error in the program. If there was en error, then the object after ERRTRAP is executed. First, 'REVAL brings the first object after the current composite (this object happens to be the second code object, that turns on the display) into the data stack and executes it. Then, the error is triggered again with ERRJMP. If there was no error, the execution goes directly to the second code object, finishing the program.

As an example another way to deal with the return stack, we will study the list processor DoL, also in library OT49. This command expects a list in level two and any object (generally a command or a function) in level one. This object is evaluated for each of the list elements in order, and the results are collected in another list, which is then returned. This program uses some things which we have not studied yet, such as loops and the Virtual Stack. You might want to skip this example now and return to it later. Here is the code, without the argument checking part:

```
1
    ::
      OVER
      >R
                        (Push list elements in return stack)
      ticR
                        (Try to get first element)
5
      NOTcaseDROP
                        (If list is empty, drop the object)
      PushVStack&Clear (Save current stack)
      GetElemBotVStack (Get first list elemement)
      BEGIN
10
        BINT1
        GetElemBotVStack
                        (Get object and evaluate)
        xEVAL
        RSWAP
```

```
ticR
                         (Get next element from list)
15
      WHILE
                         (Repeat while there are elements)
        RSWAP
      REPEAT
      DEPTH
       { }N
                         (Collect results)
20
      PopVStackAbove
                         (Get saved stack)
       4UNROLL3DROP
                         (Drop arguments & first object)
    ;
```

This program may be somewhat difficult to understand at first, but it manipulates very cleverly the return stack.

It starts by using  $>\mathbb{R}$  to insert a pointer to the list contents in the return stack. If they were not removed later, then after this program finished, each of the objects in the list would be evaluated.

Then, the first object from the list is retrieved, with ticr. This also advances the pointer in the return stack to point to the second element. If the list was empty, then ticr returns FALSE. In this case, the object to be evaluated is dropped, and the empty list remains as the result of the program.

The real fun starts when there is at least one element. The whole stack is saved as a virtual stack level, but the first element of the list (retrieved with ticR previously) is retrived into the "new" stack.

Then, a loop is started. The loop used is very similar to a User RPL WHILE...REPEAT...END loop. For more details, see Chapter 21. The object is retrieved and evaluated, and then the next element from the list is retrieved with ticr. However, since the word BEGIN pushed something in the return stack (for an explanation, see section 21.1.1), it is necessary to use RSWAP to bring the pointer to the list elements back in the first return stack level, thus allowing ticr to get one of the elements. If there was an element, RSWAP is executed again to put the return stack back into its original stack, and the loop begins again, executing the object, and so on. When there are no more elements, control goes to after the REPEAT word. All results are collected in a list, and we retrieve the saved stack above the list with the results. Then the program simply drops the original list, the object to be evaluated and the first object of the list, which were in the stack when it was pushed into the Virtual Stack.

As you have seen, this program used the return stack as a storage place; the composite that was pushed there was never executed, because each of its elements were removed until there was nothing more to execute.

Addr.	Name	Description
06E8E	NOP	$(\  ightarrow\ )$
		Does nothing.
06EEB	'R	( $ ightarrow$ 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	$( \rightarrow ? )$
		Does 'R then EVAL.
36A27	'R'R	( $ ightarrow$ ob1 ob2 )
		Does 'R twice.
34BEF	ticR	( $ ightarrow$ ob T )
		( $ ightarrow$ 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.
36A4A	'RRDROP	$( \rightarrow ob )$
		Does 'R, then RDROP.
06F9F	>R	$(::\to)$
		Pushes :: to top of return stack (skips prolog, i.e., the composite will be executed automatically).
0701F	R>	$( \rightarrow :: )$
		Creates and pops a secondary from top return stack body to stack.
07012	R@	$( \rightarrow :: )$
		Like R>, but the return stack is not popped.
0716B	IDUP	$(\  ightarrow\ )$
		Pushes top body into return stack.
06F8E	EVAL	$( ob \rightarrow ? )$
		Evaluates object.
262FB	COMPEVAL	$(comp \rightarrow ?)$
		EVAL just pushes a list back, this one executes it.
34BAB	2@REVAL	$( \rightarrow ? )$
		EVAL first object in the stream above the previous
		one.

Addr.	Name	Description
34BBB	3@REVAL	( → ? )
		EVAL first object in the stream above the stream
		above the previous one.
34A31	GOTO	$( \rightarrow )$
		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.
34A46	?GOTO	$(flag \rightarrow )$
		If TRUE, jumps, else skips five nibbles.
34A59	NOT?GOTO	$(flag \rightarrow )$
		If FALSE jumps, else skips five nibbles.
26111	RDUP	$(\rightarrow)$
		Duplicates top return stack level.
06FB7	RDROP	$(\rightarrow)$
		Pops the return stack.
343E1	2RDROP	$(\rightarrow)$
		Pops two return stack levels.
343F3	3RDROP	$(\rightarrow)$
		Pops three return stack levels.
36342	DROPRDROP	$( ob \rightarrow )$
		Does DROP then RDROP.
3597F	RDROPCOLA	$(\rightarrow)$
		Does RDROP then COLA.
34144	RSWAP	$( \rightarrow )$
		Swap in the return stack.
368C9	RSKIP	$(\rightarrow)$
		Skips first object in the return stack (i.e., the first
	()	object in the composite above this one).
2B8BE	(OBJ>R)	$( ob \rightarrow )$
		Pushes an object into the return stack, for example
		for temporary storage. If ob is a list, the list is put
		as a whole onto the stream, not the individual ele-
0-0-5	(= : = = = )	ments.
2B8E6	(R>OBJ)	$(\rightarrow \text{ob})$
0010-	G7147	Gets an object from the return stack.
0312B	SEMI	$(\rightarrow)$
		DROP the rest of the current stream.

## 19.4.1 Quoting Objects

Addr.	Name	Description
06E97	1	( $\rightarrow$ nob (nextob) )
		Pushes next object in the stream to the stack
		(skipping it).
3696E	DUP'	( ob $\rightarrow$ ob nob )
		Does DUP then '.
36996	DROP'	( ob $\rightarrow$ nob )
		Does DROP then '.
36982	SWAP'	( ob1 ob2 $ ightarrow$ ob2 ob1 nob )
		Does SWAP then '.
369AA	OVER'	( ob1 ob2 $ ightarrow$ ob1 ob2 ob1 nob )
		Does OVER then '.
369BE	STO'	( ob id/lam $\rightarrow$ nob )
		Does STO then '.
369D2	TRUE'	( $\rightarrow$ T nob )
		Pushes TRUE and the next object to the stack.
369FF	FALSE'	( $ ightarrow$ F nob )
		Pushes FALSE and the next object to the stack.
369E6	ONEFALSE'	( $ ightarrow$ #1 F nob )
		Pushes ONE, FALSE and the next object to the
		stack.
36A13	#1+'	$( \# \rightarrow \#+1 \text{ nob })$
		Does #1+ then '.
36306	'NOP	( $ ightarrow$ NOP )
		Pushes NOP to the stack.
3619E	'ERRJMP	( $ ightarrow$ ERRJMP )
		Pushes ERRJMP to the stack.
2B90B	'DROPFALSE	( $ ightarrow$ DROPFALSE )
		Pushes DROPFALSE to the stack.
25E6A	'DoBadKey	( $ ightarrow$ DoBadKey )
	-	Pushes DoBadKey to the stack.
25E6B	'DoBadKeyT	( → DoBadKey T )
	_	Pushes DoBadKey and TRUE to the stack.
2F32E	DROPDEADTRUE	( ob $\rightarrow$ DoBadKey T )
		Makes the user drop dead, then pushes TRUE.
36BBE	('x*)	$(\rightarrow x^*)$
		Pushes x* (User word *) to the stack.

Addr.	Name	Description
36BD2	'xDER	$( \rightarrow \text{xDER} )$
		Pushes xDER (User word $\partial$ ) to the stack.
27B43	'IDFUNCTION	( $ ightarrow$ xFUNCTION )
		Pushes xfunction (User word function) to
		the stack.
27B6B	'IDPOLAR	( $ ightarrow$ xPOLAR )
		Pushes XPOLAR (User word POLAR) to the stack.
27B7F	'IDPARAMETER	( $ ightarrow$ xPARAMETRIC )
		Pushes xPARAMETRIC (user word PARAMETRIC)
		to the stack.
29ED0	'Rapndit	( meta oblob4 $\rightarrow$ meta&ob oblob4
		)
		Takes ob from runstream and appends it to the
		meta starting in level 5.
36AA4	'xDEREQ	( ob $\rightarrow$ flag )
		Is ob eq to user command xDER?

## 19.4.2 Skipping Objects

Addr.	Name	Description
06FD1	COLA	Evals next obj and
		drops rest of this stream.
36A63	ONECOLA	Does ONE, then COLA.
3635B	SWAPCOLA	Does SWAP, then COLA.
3636F	XYZ>ZCOLA	Does UNROT2DROP, then COLA.
34AD3	COLA_EVAL	Returns and evals first obj
		in previous stream.
35994	COLACOLA	Drops rest of current stream
		does COLA in the above one.
0714D	SKIP	Skips 1 obj in the runstream.
35715	skipcola	Does SKIP, then COLA.
3570C	2skipcola	Does 2SKIP, then COLA.
35703	3skipcola	Does 3SKIP, then COLA.
356D5	5skipcola	Skips 5 objects, then does COLA.
363FB	COLASKIP	Drops rest of current stream
		and skips one obj in above stream.

## Chapter 20 Conditionals

In System RPL, conditionals are a bit different from User RPL. The first difference is that in User RPL, a "false" is represented by the real number zero; any other value represents a "true". In System RPL, a "false" is represented by the word FALSE, and a "true" is represented by the word TRUE (amazing!). These words just put themselves in the stack when run. All commands that do a test return one of them. Words like IT or case take one of them as argument.

Should you need, you can convert a TRUE or FALSE to a (real) 0 or 1 with COERCEFLAG. There is not a dedicated function to do the opposite transformation, like UNCOERCEFLAG, but %0<> does the job perfectly.

There are many commands that put TRUE, FALSE, or some combination of them in the stack. See the list below.

The Boolean operators are present, too: NOT, AND, OR and XOR. There are some combinations, see below for a list.

#### **20.1 Tests**

The test words are commands which take one or more arguments and return either TRUE or FALSE, after doing some kind of comparison between the arguments. The tests for each kind of object type are listed in the reference section of the chapter of each object type. Tests for object type can be found on Chapter 29. Other kinds of tests are listed in the reference section below.

The most important of these tests are EQ and EQUAL. Both take two objects and return a flag. The first checks if the objects are the same, i.e., occupy the same address in memory. The second checks if the objects are equal in terms of contents. The difference is that :: BINT2 # 2 EQUAL; returns TRUE, but if EQUAL is replace by EQ, then the program returns FALSE, because one object is the built-in bint 2, found at address #3311B, and the other is a bint whose address cannot be predicted, but certainly is not in the ROM.

Another example: if you put a string in level one, and press ENTER, EQ

20.3. Case 133

and EQUAL will return TRUE. However, if you enter a string, and then enter again the exact same string, only EQUAL will return TRUE. This happens because the contents of the strings are the same, but they are different objects in memory, occupying each a different address in memory. They just happen to have the same contents. When possible, you should use EQ in your programs since it is faster than EQUAL.

#### 20.2 **If...Then...Else**

Most of the time, you will create this kind of conditionals with the IT and ITE commands:

#### Word Stack and Action

IT (flag  $\rightarrow$ )

If the flag is TRUE, the next object is executed, otherwise it is skipped.

ITE (flag  $\rightarrow$ )

If the flag is TRUE, the next object is executed, and the second is skipped. If it is FALSE, the next object is skipped and the second is executed.

The following snippet changes a zero into a one, but does nothing to other numbers:

```
... DUP %0= IT %1+ ...
```

The code below will output "Equal" if two objects are equal, and "Not equal" if not:

```
... EQUAL $ "Equal" $ "Not equal" ...
```

Naturally, when you need to execute several commands, you will need to include them in a secondary.

#### 20.3 Case

The CASE words are a combination of IT, SKIP and COLA (see Chapter 19). The basic word is case, but there are combinations of it with tests and other commands.

case takes a flag in level one. If the flag is TRUE, the next object is

134 20. Conditionals

executed, but only it: the rest of the stream is dropped. So, TRUE case is equivalent to COLA. If the flag is FALSE, the next object is skipped and execution continues after it. So, FALSE case is the same as SKIP.

The example below shows how to build a familar case structure similar to that found in other languages (even User RPL!). It outputs a string representing the bint in level one.

There are many words that combine case with other words. One of them is OVER#=case. It is not difficult to figure out what it does: first, OVER. Then, #= compares two bints. Finally, the case works as before. Using this word, the code above could be rewritten as:

```
1 ::
BINTO OVER#=case $ "Zero"
BINT1 OVER#=case $ "One"
BINT2 OVER#=case $ "Two"

5 ...
```

In the reference section below, you will find a list of the words that execute a case besides some other action. These words are composed of an initial part, the case itself and a final part. Some have only the initial or final part besides the case, some have both. The initial part represents the commands that are executed before the case, and it should be pretty straightforward to understand their action, as an example NOTcase is equivalent to NOT followed by case. For the final part, things become more complicated, because there are two kinds of final part. The first kind has the final part written in UPPER-CASE letters. The commands in the final kind are executed if the test is true. You only need to provide the action for the FALSE situation. For example, this snippet

```
... caseDROP <FalseAction> ...
is equivalent to
... case DROP <FalseAction> ...
```

The second type comprises the words that have the final part in lowercase letters. In this case, the commands in the final part are executed *along* with the object that follows case when the test is true. In other words, this snippet

```
... casedrop <TrueAction> <FalseAction> ...
is equivalent to
... case :: DROP <TrueAction> ; <FalseAction> ...
```

Unfortunately, some entries have been misnamed, and this convention was not adhered. These entries are marked clearly in the descriptions below.

Also, the "stack diagrams" of most of the words below are not true stack diagrams. What is on the left side of the arrow is the contents of the stack before calling the entry, as usual. ob1 and ob2 are different objects. f1 and f2 are different flags; T represents TRUE and F, FALSE. #m and #n represent two binary integers, #m being smaller than #n. #set is the number of a flag, and this flag is set, #clr is the number of a flag, this flag being clear. On the right of the arrow, the objects which will be executed when the stack matches the left side of the arrow are represented. The initial stream has the form: :: <test\_word> <obl> ... <obn> ; In the diagrams, <rest> represents all the objects after the object that appers before <rest>. In this right side of the arrow there are also objects appearing without the angle brackets already. These are objects in the data stack that result after the word is run, and not objects in the runstream.

#### 20.4 Reference

#### 20.4.1 Boolean Flags

Addr.	Name	Description
2602B	COERCEFLAG	( T $\rightarrow$ %1 )
		$(F \rightarrow \$0)$
		Converts system flag to user flag, drops current
		stream.
301BA	%0<>	( % $ ightarrow$ flag )
		Can be used to change a user flag into a system
		flag.
03A81	TRUE	$( \rightarrow T )$
27E87	TrueTrue	( $ ightarrow$ T T )

20. Conditionals

Addr.	Name	Description
36540	TrueFalse	( $\rightarrow$ T F )
		aka: TRUEFALSE
03AC0	FALSE	( $ ightarrow$ F )
36554	FalseTrue	( $ ightarrow$ F T )
		aka: FALSETRUE
283E8	FalseFalse	$( \rightarrow F F )$
27E9B	failed	$( \rightarrow F T )$
35280	DROPTRUE	$( ob \rightarrow T )$
2D7006	^2DROPTRUE	(ob ob' $\rightarrow$ T)
35289	DROPFALSE	$( ob \rightarrow F )$
35B32	2DROPFALSE	( ob1 ob2 $\rightarrow$ F )
28211	NDROPFALSE	( oblobn $\#n \rightarrow F$ )
2812F	SWAPTRUE	( ob1 ob2 $\rightarrow$ ob2 ob1 T )
374BE	SWAPDROPTRUE	( ob1 ob2 $\rightarrow$ ob2 T )
35EF2	XYZ>ZTRUE	( ob1 ob2 ob3 $\rightarrow$ ob3 T )
2962A	RDROPFALSE	$( \rightarrow F )$
		Puts FALSE in the stack and drops rest of cur-
		rent stream.
03AF2	NOT	( flag $ ightarrow$ flag' )
		Returns FALSE if the input is TRUE, and vice-
		versa.
03B46	AND	( flag1 flag2 $\rightarrow$ flag )
		Returns TRUE if both flags are TRUE.
03B75	OR	( flag1 flag2 $\rightarrow$ flag )
		Returns TRUE if either flag is TRUE.
03ADA	XOR	( flag1 flag2 $\rightarrow$ flag )
		Returns TRUE if flags are different.
365F9	ORNOT	(flag1 flag2 $\rightarrow$ flag)
		Returns FALSE if either flag is TRUE.
35C7C	NOTAND	( flag1 flag2 → flag )
		Returns TRUE if flag1 is TRUE and flag2 is
		FALSE.
35CB8	ROTAND	(flag1 ob flag2 → ob flag)
		Returns TRUE if either flag is TRUE.

#### 20.4.2 General Tests

Addr.	Name	Description
03B2E	EQ	( ob1 ob2 $\rightarrow$ 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.
36621	2DUPEQ	( ob1 ob2 $ ightarrow$ ob1 ob2 flag )
		Does 2DUP then EQ.
3664E	EQOR	( flag ob1 ob2 $\rightarrow$ flag' )
		Does EQ then OR.
3607F	EQOVER	( ob3 ob1 ob2 $\rightarrow$ ob3 flag ob3 )
		Does EQ then OVER.
3663A	EQ:	( ob $\rightarrow$ flag )
		EQ with the next object in the current stream.
36635	DUPEQ:	$( ob \rightarrow ob flag )$
		Does DUP then EQ:.
03B97	EQUAL	$( ob1 ob2 \rightarrow flag )$
		Returns TRUE if the objects are equal (but not nec-
		essarily the same), i.e., their prologs and contents
		are the same.
3660D	EQUALNOT	$( ob1 ob2 \rightarrow flag )$
	~	Returns TRUE if the objects are different.
36662	EQUALOR	( flag ob1 ob2 → flag' )
	~	Does EQUAL then OR.
0FF006	^Contains?	( ob1 ob2 $\rightarrow$ ob1 ob2 flag )
		Tests if ob1 contains ob2. If ob1 is a symbolic then
		ob1 is searched for embedded ob2. If ob1 is a list
		then ob1 is traversed for a direct match. Other-
		wise, tests if ob1 and ob2 are equal.
		,

#### 20.4.3 True/False Tests

Addr.	Name	Description
34AA1	?SEMI	( T $\rightarrow$ :: ; )
		( F $ ightarrow$ :: <obl> <rest> ; )</rest></obl>
34A92	NOT?SEMI	( T $ ightarrow$ :: <obl> <rest> ; )</rest></obl>
		( F $ ightarrow$ :: ; )

138 20. Conditionals

Addr.	Name	Description
3692D	?SEMIDROP	( ob T $\rightarrow$ :: ob ; )
		( ob F $\rightarrow$ :: <ob1> <rest> ; )</rest></ob1>
34BD8	NOT?DROP	( ob T $\rightarrow$ :: ob $<$ ob1 $>$ $<$ rest $>$ ; )
		$( \ ob \ F \ \to \ \vcentcolon : \ < ob1 > \ < rest > \ ; \ )$
35F56	?SWAP	( ob1 ob2 T $\rightarrow$ :: ob2 ob1 <ob1></ob1>
		<rest> ; )</rest>
		( ob1 ob2 F $ ightarrow$ :: ob1 ob2 <ob1></ob1>
2 5 5 5 7		<pre><rest> ; ) (</rest></pre>
35DDA	?SKIPSWAP	( ob1 ob2 T $ ightarrow$ :: ob1 ob2 <ob1> <rest> ; )</rest></ob1>
		( ob1 ob2 F $\rightarrow$ :: ob2 ob1 <ob1></ob1>
		( OD1 OD2 F → OD2 OD1 <od1> &lt; rest&gt; ; )</od1>
35F97	?SWAPDROP	( obl ob2 T $\rightarrow$ :: obl <obl> <rest> ;</rest></obl>
33177	: SWAF DROF	)
		( obl ob2 F $\rightarrow$ :: ob2 <obl> <rest> ;</rest></obl>
		)
35F7E	NOT?SWAPDROP	( ob1 ob2 T $\rightarrow$ :: ob2 <ob1> <rest> ;</rest></ob1>
		)
		( ob1 ob2 F $ ightarrow$ :: ob1 <ob1> <rest> ;</rest></ob1>
		)
070FD	RPIT	( T ob $\rightarrow$ :: ob $<$ ob1 $>$ $<$ rest $>$ ; )
		( F ob $\rightarrow$ :: $\langle \text{ob1} \rangle \langle \text{rest} \rangle$ ; )
		ob is actually executed, and not pushed in the
		stack.
070C3	RPITE	( T ob1 ob2 $\rightarrow$ :: ob1 <ob1> <rest> ;</rest></ob1>
		$( F ob1 ob2 \rightarrow ob2 < ob1> < rest> ; )$
		ob1 or ob2 is actually executed, and not pushed in the stack.
34AF4	COLARPITE	in the stack. ( T obl ob2 $\rightarrow$ :: obl ; )
34AF4	COLARPITE	$( \text{F ob1 ob2} \rightarrow \cdots \text{ob1 } )$ $( \text{F ob1 ob2} \rightarrow \cdots \text{ob2 } ; )$
		ob1 or ob2 is actually executed, and not pushed
		in the stack.
34B4F	2'RCOLARPITE	Return to composite and ITE there.
34A22		$(T \rightarrow ::  < rest> ;)$
		( F $\rightarrow$ :: <ob2> <rest> ; )</rest></ob2>
0712A	?SKIP	( T $\rightarrow$ :: <ob2> <rest> ; )</rest></ob2>
		( F $ ightarrow$ :: <obl> <rest> ; )</rest></obl>
		aka: NOT_IT

Addr.	Name	Description
34B3E	ITE	( T $\rightarrow$ :: <ob1> <ob3> <rest> ; )</rest></ob3></ob1>
		( F $\rightarrow$ :: <ob2> <rest> ; )</rest></ob2>
36865	COLAITE	( T $\rightarrow$ :: $\langle ob1 \rangle$ ; )
		( F $\rightarrow$ :: $\langle ob2 \rangle$ ; )
34ABE	ITE_DROP	( ob T $\rightarrow$ :: <ob2> <rest> ; )</rest></ob2>
		( ob F $\rightarrow$ :: ob $<$ ob1 $>$ $<$ rest $>$ ; )
36EED	ANDITE	( f1 f2 $\rightarrow$ :: <ob1> <ob3> <rest> ; )</rest></ob3></ob1>
		$( f1 f2 \rightarrow ::   ; )$
349F9	case	$( T \rightarrow :: < ob1 > ; )$
		$( F \rightarrow ::   ; )$
34A13	NOTcase	$( T \rightarrow :: < ob2 > < rest > ; )$
		$(F \rightarrow :: < ob1 > ;)$
36D4E	ANDcase	$( f1 f2 \rightarrow :: < ob1 > ; )$
		$( f1 f2 \rightarrow :: < ob2 > < rest > ; )$
36E6B	ANDNOTcase	$( f1  f2  \rightarrow ::                              $
		$( f1 f2 \rightarrow :: \langle ob2 \rangle \langle rest \rangle ; )$
359E3	ORcase	$( f1 f2 \rightarrow :: \langle ob1 \rangle ; )$
	-	$( f1 f2 \rightarrow :: \langle ob2 \rangle \langle rest \rangle ; )$
3495D	casedrop	$( \ ob \ \mathtt{T} \ \to \ :: \ \ < ob1 > \ ; \ )$
2424-		( ob F $\rightarrow$ :: ob <ob2> <rest> ; )</rest></ob2>
3494E	NOTcasedrop	$( \text{ ob } T \rightarrow :: \text{ ob } <\text{ob2}> <\text{rest}> ; )$
24005	0 -1	$(ob F \rightarrow :: \langle ob1 \rangle ;)$
34985	case2drop	$( ob1 ob2 T \rightarrow :: ; )$
		( ob1 ob2 F $\rightarrow$ :: ob1 ob2 <ob2></ob2>
34976	MOTTenandamen	<rest> ; ) ( ob1 ob2 T <math> ightarrow</math> :: ob1 ob2 <ob2></ob2></rest>
34970	Norcasezdrop	<pre>( obl ob2 l → ·· obl ob2 <ob2> <rest> ; )</rest></ob2></pre>
		( ob1 ob2 F $\rightarrow$ :: <ob1> ; )</ob1>
3/10p1	caseDROP	$( \text{ ob } T \rightarrow :: ; )$
34901	CaseDNOF	$( \text{ ob } F \rightarrow :: \text{ ob } < \text{obl} > < \text{rest} > ; )$
349C6	NOTcaseDROP	$( ob T \rightarrow :: ob < obl > < rest > i )$ $( ob T \rightarrow :: ob < obl > < rest > i )$
34700	NOICASCDROF	$\begin{array}{cccccccccccccccccccccccccccccccccccc$
368FB	casedrptru	$( b T \rightarrow T )$
3001 B	cascarpera	( ob F $\rightarrow$ :: ob <obl> <rest> ; )</rest></obl>
		Note: should be caseDRPTRU.
365B3	casedrpfls	( ob T $\rightarrow$ F )
30020	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	( ob F $\rightarrow$ :: ob <ob1> <rest> ; )</rest></ob1>
		Note: should be caseDRPFLS.

140 20. Conditionals

```
Addr. Name
                          Description
36B3A
       NOTcsdrpfls
                          ( ob T \rightarrow :: ob <ob1> <rest> ; )
                           ( ob F \rightarrow F )
                          Note: should be NOTcaseDRPFLS.
                          ( ob1 ob2 T \rightarrow :: ; )
349D6 case2DROP
                           ( ob1 ob2 F \rightarrow :: ob1 ob2 <ob1>
                           <rest> ; )
349EA NOTcase2DROP
                          ( ob1 ob2 T \rightarrow :: ob1 ob2 <ob1>
                           <rest> ; )
                           ( ob1 ob2 F \rightarrow :: ; )
365CC case2drpfls
                          ( ob1 ob2 T \rightarrow F )
                          ( ob1 ob2 F \rightarrow :: ob1 ob2 <ob1>
                           <rest> ; )
                          Note: should be case2DRPFLS.
                          (T \rightarrow T)
3652C caseTRUE
                          ( F \rightarrow :: <ob1> <rest> ; )
36914 NOTcaseTRUE
                          ( T \rightarrow :: <ob1> <rest> ; )
                          ( F \rightarrow T )
                          (T \rightarrow F)
365E5 caseFALSE
                           ( F \rightarrow :: <ob1> <rest> ; )
2B2C5 NOTcaseFALSE
                          ( T \rightarrow :: <obl> <rest> ; )
                          ( F \rightarrow F )
359AD COLAcase
                          ( T \rightarrow :: \langle ob1 \rangle ; )
                           ( F \rightarrow :: <ob2> <rest> ; )
                          Drops the rest of current stream and executes
                           case in the stream above.
                          ( T \rightarrow :: <ob2> <rest> ; )
359C8 COLANOTcase
                           ( F \rightarrow :: < ob1 > ; )
                           Drops the rest of current stream and executes
                          NOTcase in the stream above.
```

## 20.4.4 Binary Integer Tests

Addr.	Name	Description
363B5	#=?SKIP	( $\#m \ \#n \rightarrow ::   ; )$
		( $\#m \ \#n \rightarrow :: < ob1 > < rest > ; )$
363E2	#>?SKIP	( $\#m \ \#n \rightarrow :: < ob1 > < rest > ; )$
		( $\#m \ \#n \rightarrow :: < ob2 > < rest > ;$ )

```
Addr.
                           Description
        Name
35C54
                           ( \#m \ \#n \rightarrow :: <ob1> <ob3> <rest> ; )
         #=ITE
                           ( \#m \#n \rightarrow :: <ob2> <rest> ; )
36F29
                           ( \#m \ \#n \rightarrow :: <ob1> <ob3> <rest> ; )
        #<ITE
                           ( \#m \ \#n \rightarrow :: < ob2 > < rest > ; )
36F3D
                           ( \#m \#n \rightarrow :: <ob2> <rest> ; )
        #>ITE
                           ( \#m \#n \rightarrow :: <ob1> <ob3> <rest> ; )
348D2
                           ( \#m \#n \rightarrow :: <ob1> ; )
        #=case
                           ( \#m \ \#n \rightarrow :: <ob2> <rest> ; )
                           ( \#m \ \#n \rightarrow :: \ \#m \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )
348E2
        OVER#=case
                           34939
        #=casedrop
                           ( \#m \#n \rightarrow :: <ob1> ; )
                           Note: should be OVER#=casedrop.
36590
        #=casedrpfls
                          ( m m m \rightarrow F)
                           Note: should be OVER#=caseDRPFLS.
        #<>case
                           ( \#m \#n \rightarrow :: <ob2> <rest> ; )
36D9E
                           ( \#m \#n \rightarrow :: < ob1 > ; )
36D76
                           ( \#m \#n \rightarrow :: <ob1> ; )
        #<case
                           ( \#m \#n \rightarrow :: <ob2> <rest> ; )
        #>case
                           ( \#m \#n \rightarrow :: <ob2> <rest> ; )
36DCB
                           ( \#m \#n \rightarrow :: < ob1 > ; )
                           ( #0 \rightarrow :: ; )
34A7E
        #0=?SEMI
                           ( \# \rightarrow :: < ob1 > < rest > ; )
                           ( \#0 \rightarrow :: <ob2> <rest> ; )
36383
        #0=?SKIP
                           ( \# \rightarrow :: < ob1 > < rest > ; )
                           ( \#0 \rightarrow :: <ob1> <ob3> <rest> ; )
36F15
        #0=ITE
                           ( \# \rightarrow :: < ob2 > < rest > )
36ED4
        DUP#0=IT
                           ( #0 \rightarrow :: #0 < ob1 > < rest > ; )
                           ( \# \rightarrow :: \# < ob2 > < rest > ; )
                           ( \#0 \rightarrow :: \#0 < ob1 > < ob3 > < rest > ; )
36F51
        DUP#0=ITE
                           ( \# \rightarrow :: \# < ob2 > < rest > ; )
                           ( \#0 \rightarrow :: < ob1 > ; )
348FC
        #0=case
                           ( \# \rightarrow :: < ob2 > < rest > ; )
348F7
        DUP#0=case
                           ( \#0 \rightarrow :: \#0 < ob1> ; )
                           ( \# \rightarrow :: \# < ob2 > < rest > ; )
3490E
       DUP#0=csedrp
                          ( #0 \rightarrow :: < ob1 > ; )
                           ( \# \rightarrow :: \# < ob2 > < rest > ; )
```

142 20. Conditionals

Addr.	Name	Description
36D21	DUP#0=csDROP	( #0 → :: ; )
		( $\#$ $\rightarrow$ :: $\#$ <obl> <rest> ; )</rest></obl>
36D8A	#1=case	( $\#1 \rightarrow :: < ob1 > ; )$
		( $\# \rightarrow :: < ob2 > < rest > ; )$
3639C	#1=?SKIP	( $\#1 \rightarrow ::   ; )$
		( $\# \rightarrow :: < ob1 > < rest > ;$ )
36DB2	#>2case	( $\#0/\#1/\#2 \rightarrow ::   ; )$
		( $\# \rightarrow :: < ob1 > ; )$
25E72	?CaseKeyDef	( $\#$ $\#$ ' $\rightarrow$ :: ' ob1 T ; )
		( $\# \ \#' \rightarrow :: < ob2 > < rest > ; )$
		Compares two bints. If equal, quotes the next
		object from the runsream and returns it along
		with TRUE.
25E73	?CaseRomptr@	$( \# \#' \rightarrow ob T )$
		$( \ \# \ \#' \  o \ F \ )$
		$( \# \#' \rightarrow :: < ob2 > < rest > ; )$
		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 di-
		rectly for key handlers.

## 20.4.5 Real and Complex Number Tests

Addr.	Name	Description
2B149	%0=case	( %0 → :: %0 <ob1> ; )</ob1>
		( ob $ ightarrow$ :: ob $\langle \text{ob2} \rangle$ $\langle \text{rest} \rangle$ ; )
36DDF	j%0=case	( %0 $\rightarrow$ :: $\langle ob1 \rangle$ ; )
		( ob $ ightarrow$ :: <ob2> <rest> ; )</rest></ob2>
2B15D	C%0=case	( C%0 $ ightarrow$ :: C%0 $ m $ ; )
		( ob $ ightarrow$ :: ob $\langle \text{ob2} \rangle$ $\langle \text{rest} \rangle$ ; )
2B11C	num0=case	( $0 \rightarrow :: 0 < ob1 > ;$ )
		( ob $ ightarrow$ :: ob $\langle \text{ob2} \rangle$ $\langle \text{rest} \rangle$ ; )
		Both a real and a complex zero are TRUE conditions
		for this test.
2B1A3	%1=case	( %1 $\rightarrow$ :: %1 $<$ ob1 $>$ ; )
		( ob $ ightarrow$ :: ob <ob2> <rest> ; )</rest></ob2>

Addr.	Name	Description
2B1C1	C%1=case	( C%1 → :: C%1 <ob1> ; )</ob1>
		( ob $\rightarrow$ :: ob $<$ ob2 $>$ $<$ rest $>$ ; )
2B176	num1=case	$(1 \rightarrow :: 1 < ob1 > ;)$
		( ob $\rightarrow$ :: ob <ob2> <rest> ; )</rest></ob2>
		Both a real and a complex one are TRUE conditions
		for this test.
2B20C	%2=case	( $%2 \rightarrow :: %2 < ob1 > ; )$
		( ob $\rightarrow$ :: ob <ob2> <rest> ; )</rest></ob2>
2B22A	C%2=case	( $C%2 \rightarrow :: C%2 < ob1 > ;$ )
		( ob $\rightarrow$ :: ob $<$ ob2 $>$ $<$ rest $>$ ; )
2B1DF	num2=case	$(2 \rightarrow :: 2 < ob1 > ;)$
		( ob $\rightarrow$ :: ob <ob2> <rest> ; )</rest></ob2>
		Both a real and a complex two are TRUE conditions
		for this test.
2B289	%-1=case	( %-1 $\rightarrow$ :: %-1 $<$ ob1 $>$ ; )
		( ob $\rightarrow$ :: ob $<$ ob2 $>$ $<$ rest $>$ ; )
2B2A7	C%-1=case	( $C\%-1 \rightarrow :: C\%-1 < ob1 > ;$ )
		( ob $ ightarrow$ ob $ m  $ ; )
2B25C	num-1=case	$(-1 \rightarrow :: -1 < ob1 > ;)$
		( ob $\rightarrow$ :: ob <ob2> <rest> ; )</rest></ob2>
		Both a real and a complex -1 are TRUE conditions
		for this test.

## 20.4.6 Meta Object Tests

Addr.	Name	Description
2AFFB	MEQ1stcase	( meta&ob1 ob2 $ ightarrow$ ob1=ob2 ? case )
		Meta&ob1 ob2 ob1=ob2? case
2AF37	AEQ1stcase	( meta&ob $ ightarrow$ ob=nob ? case )
		Meta&ob ob=nob? case
2B01B	MEQopscase	( metal&ob1 meta2&ob2 ob3 $ ightarrow$ )
		Meta1&ob1 Meta2&ob2 ob3
2B06A	AEQopscase	meta1&ob1 meta2&ob2
		Meta1&ob1 Meta2&ob2
2B083	Mid1stcase	( meta&ob $ ightarrow$ ob is id )
		lam?case
		Meta&ob ob is id or lam? case

144 20. Conditionals

Addr.	Name	Description
2AE32	M-1stcasechs	( Meta&NEG $ ightarrow$ Meta COLA )
		( Meta $ ightarrow$ Meta SKIP )
		( Meta&( $%<0$ ) $\rightarrow$ Meta&ABS( $%$ ) COLA )
		Meta&NEG Meta COLA ; Meta Meta SKIP
		Meta&(%<0) $Meta&ABS(%)$ COLA

## 20.4.7 General Object Tests

Addr.	Name	Description
36EBB	EQIT	( ob1 ob1 $\rightarrow$ :: <ob1> <rest> ; )</rest></ob1>
		( ob1 ob2 $ ightarrow$ :: <ob2> <rest> ; )</rest></ob2>
36F01	EQITE	( ob1 ob1 $ ightarrow$ :: <ob1> <ob3> <rest></rest></ob3></ob1>
		; )
		( ob1 ob2 $ ightarrow$ :: <ob2> <rest> ; )</rest></ob2>
36D3A	jEQcase	( ob1 ob1 $ ightarrow$ :: $\langle \text{ob1}  angle$ ; )
		( ob1 ob2 $ ightarrow$ :: <ob2> <rest> ; )</rest></ob2>
34999	EQcase	( ob1 ob1 $ ightarrow$ :: ob1 $ m $ ; )
		( ob1 ob2 $\rightarrow$ :: ob1 <ob2> <rest> ;</rest></ob2>
		)
		Note: Should be called OVEREQuase.
359F7	REQcase	( ob $\rightarrow$ :: ob $<$ ob2 $>$ ; )
		( ob $\rightarrow$ :: ob <ob3> <rest> ; )</rest></ob3>
		EQcase with the next object in the runstream.
34920	EQcasedrop	( ob1 ob1 $\rightarrow$ :: <ob1> ; )</ob1>
		( ob1 ob2 $\rightarrow$ :: ob1 <ob2> <rest> ;</rest></ob2>
		) N + 1 111
05-40		Note: should be OVEREQuasedrop.
35A10	REQcasedrop	$( ob \rightarrow \langle ob2 \rangle ; )$
		$( ob \rightarrow   ; )$
		EQcasedrop with the next object in the run-
26760		stream.
36D62	EQUALcase	$( ob1 ob1 \rightarrow :: ; )$
26888		$( ob1 ob2 \rightarrow ::   ; )$
36E7F	EQUALNOTcase	$( ob1 ob1 \rightarrow ::   ; )$
36500	EOUNT de de des	$( ob1 ob2 \rightarrow :: ; )$
36D08	EQUALcasedrp	( ob obl ob2 $\rightarrow$ :: $\langle ob1 \rangle$ ; )
		( ob ob1 ob2 $\rightarrow$ :: ob <ob2> <rest></rest></ob2>
		; )

```
Addr.
                               Description
         Name
         EQUALcasedrop
                               ( ob1 ob2 \rightarrow :: <ob1> ; )
2AD81
                               ( ob1 ob2 \rightarrow :: ob1 <ob2> <rest> ;
                               )
29E99
         tok=casedrop
                               ( \$ \$' \rightarrow :: < ob1 > ; )
                               ( \$ \$' \rightarrow :: \$ < ob2 > < rest > ; )
                               Note: should be OVERtok=casedrop.
                               ( seco \rightarrow :: seco < ob2 > < rest > ; )
2ADBD
         nonopcase
                               ( ob \rightarrow :: ob \langle ob1 \rangle ; )
                               ( id \rightarrow :: id \langle ob1 \rangle ; )
2B0CC
         idntcase
                               ( ob \rightarrow :: ob <ob2> <rest> ; )
36E93
         dIDNTNcase
                               ( id \rightarrow :: id \langle ob2 \rangle \langle rest \rangle ; )
                               ( ob \rightarrow :: ob \langle ob1 \rangle ; )
2B0EF
         idntlamcase
                               ( id/lam \rightarrow :: id < ob1 > ; )
                               ( ob \rightarrow :: ob <ob2> <rest> ; )
                               ( % \rightarrow :: < ob1 > ; )
36DF3
         REALcase
                               ( ob \rightarrow :: <ob2> <rest> ; )
                               ( % \rightarrow :: % < ob2 > < rest > ; )
36EA7
         dREALNcase
                               ( ob \rightarrow :: ob <ob1> ; )
36E07
         dARRYcase
                               ( [] \rightarrow :: [] \langle ob1 \rangle; )
                               ( ob \rightarrow :: ob < ob2 > < rest > ; )
                               \{\} \rightarrow :: \{\} \text{ ob1 } ; \}
36E43
         dLISTcase
                               ( ob \rightarrow :: ob < ob2 > < rest > ; )
                               \{\} \rightarrow :: \{\} < ob2 > < rest > ; \}
260C6
        NOTLISTcase
                               ( ob \rightarrow :: ob <ob1> ; )
                               ( seco \rightarrow :: seco < ob2 > < rest > ; )
260D0
         NOTSECOcase
                               ( ob \rightarrow :: ob \langle ob1 \rangle ; )
260CB
        NOTROMPcase
                               ( romp \rightarrow :: romp < ob2 > < rest > ; )
                               ( ob \rightarrow :: ob \langle ob1 \rangle ; )
                               ( %/C%/[]/[L] \rightarrow :: <ob1> ; )
2ADE0 numb1stcase
                               ( ob \rightarrow :: ob2 <rest> ; )
                               If \%, C\%, [ ] or [L] then COLA, else SKIP.
```

146 20. Conditionals

## 20.4.8 Miscellaneous

Addr.	Name	Description
36F65	UserITE	( $\#set \rightarrow ::    ; )$
		( $\#clr \rightarrow ::  < rest> ; )$
36F79	SysITE	( $\#set \rightarrow ::    ; )$
		( $\#clr \rightarrow ::  < rest> ; )$
36C4F	caseDoBadKey	( T $\rightarrow$ :: DoBadKey ; )
		( F $ ightarrow$ :: <obl> <rest> ; )</rest></obl>
		aka: caseDEADKEY
36C36	caseDrpBadKy	( ob T $ ightarrow$ :: DoBadKey $;$ )
		( ob F $ ightarrow$ :: ob <obl> <rest> ; )</rest></obl>
361B2	caseERRJMP	( T $\rightarrow$ :: ERRJMP ; )
		( F $\rightarrow$ :: <ob> <rest> ; )</rest></ob>
36B53	caseSIZEERR	( T $ ightarrow$ :: SIZEERR ; )
		( F $\rightarrow$ :: <ob> <rest> ; )</rest></ob>
36B67	NcaseSIZEERR	( T $\rightarrow$ :: <ob> <rest> ; )</rest></ob>
		( F $ ightarrow$ :: SIZEERR ; )
36BAA	NcaseTYPEERR	( T $\rightarrow$ :: <ob1> <rest> ; )</rest></ob1>
		( F $ ightarrow$ :: TYPEERR ; )
25EEE	NoEdit?case	( $\rightarrow$ :: <obl> <rest> ; )</rest></obl>
		$( \rightarrow :: < rest > ; )$
		Tests if there is no edit line active.
36E57	EditExstCase	( $\rightarrow$ :: <obl> <rest> ; )</rest></obl>
		$( \rightarrow :: < rest > ; )$
		Tests if there is an edit line active.
2BE36	(ALGcase)	( $\rightarrow$ :: <ob1> ; )</ob1>
		( $\rightarrow$ :: <ob2> <rest> )</rest></ob2>
		Tests for algebraic mode and does case.

# Chapter 21 Loops

As in User RPL, there are two types of loops in System RPL: indefinite loops and definite loops. Indefinite loops are loops in which you do not know beforehand how many times it will be executed: it will repeat until a specific condition is met. They are created in a very similar manner to User RPL indefinite loops. Definite loops, on the other hand, are executed a number of times specified before its start. They not created exactly like in User RPL, but their use is simple and more powerful. For example, you can change the number of times to run the loop while running it.

In the descriptions below, the elements between < > can consist of several objects, unless otherwise noted.

#### 21.1 Indefinite Loops

In System RPL, indefinite loops can be made in three ways. The first is the WHILE loop. It is created like this:

This kind of loop executes <test clause>, and if the test is TRUE, <loop object> is executed, and the loop starts again. If the test returned FALSE, then execution resumes past REPEAT. If the first test returned FALSE, this loop would never be executed.

This loop requires <loop object> to be a single object. Most of the times, this will be a composite.

The second type of indefinite loop is the  ${\tt UNTIL}$  loop. It is created like this:

148 21. Loops

```
1 BEGIN <loop clause>
```

This loop is always executed at least once. The word UNTIL expects a flag. If it is FALSE, the <loop clause> is executed again. If it is TRUE, execution continues past UNTIL.

There is also a third type of indefinite loop:

```
1 BEGIN <loop object> AGAIN
```

This loop has no test. To exit it, an error condition must happen, or the return stack must be directly manipulated. This is useful if the loop code contains several different locations at which decisions about repeating or exiting the loop have to be made.

#### 21.1.1 How Indefinite Loops Work

Indefinite loops are formed by combinations the words BEGIN, WHILE, REPEAT, UNTIL and AGAIN. These have nothing special, they are commands just like the others, that when combined allow loops to be made. They work by manipulating the runstream and the return stack, so be sure you understand this concepts (see section 19.1 if in doubt).

Stack and action
$(\rightarrow)$
This copies the interpreter pointer into the return stack.
(flag $\rightarrow$ )
If the flag is TRUE, pops the return stack, otherwise sets the inter-
preter pointer to the topmost address of the return stack, without
popping it.
(flag $\rightarrow$ )
If the flag is TRUE, does nothing. Otherwise, pops the return stack
and skips the next two objects in the runstream.
$(\rightarrow)$
Sets the interpreter pointer to the topmost pointer of the return stack, without popping it.

#### Word Stack and action

AGAIN (

Sets the interpreter pointer to the topmost address of the return stack, without popping it.

From the descriptions above, you should have understood how the loops work, and also why the BEGIN...WHILE...REPEAT loops requires a single object between WHILE and REPEAT.

#### 21.2 Definite Loops

Definite loops are created with DO and LOOP (or other equivalent words). DO takes two bints from the stack, representing the stop and start values. The start value is stored as the current index, which can be recalled with INDEX@. The stop value can be recalled with ISTOP@. You can store a new value to one of them with INDEXSTO and ISTOPSTO, respectively.

DO's counterparts are LOOP and +LOOP. The former increments the index value by one, and checks if the new value is greater than or equal to the stop value, exiting the loop if it is. If not, the loop is executed again. +LOOP works similarly, incrementing the index by the bint in level one.

The standard form of a DO loop is

```
stop start DO <loop clause> LOOP
```

which executes <loop clause> for each index value from start to stop-1. Note that the stop value is greater than what it would be in User RPL, so pay attention. Also, the "stop" value comes before the "start" value.

There are several words provided to be used with DO loops, like ONE\_DO. They are listed below.

Here is an example of a simple loop which outputs the bints #1h, #2h, #3h and #4h to the stack:

```
1 ::
BINT5 BINT1
DO
INDEX@
5 LOOP
;
```

It could be changed to:

150 21. Loops

```
1 ::
BINT5 ONE_DO
INDEX@
LOOP
5 ;
```

#### 21.2.1 How a DO Loop Works

If you have some familiarity with concepts such as the return stack and the runstream (described in section 19.1), this section will explain to you how a DO loop works.

When the word DO is executed, it pushes the interpreter pointer (which points to the first object after the DO) to the return stack. It also creates a Doloop environment, storing the initial and stop values.

Execution continues normally, running all commands between DO and  $\ensuremath{\mathtt{LOOP}}.$ 

When LOOP is executed, it increments the current value in the most recent Doloop environment. If it is greater than or equal to the stop value of that environment, the environment is destroyed, and one level is popped out of the return stack. This removes the pointer to the first object after DO, and execution continues normally after LOOP. If the value is smaller, then the interpreter pointer is set to the top value in the return stack, causing the execution to re-start at the first object after the DO.

#### 21.3 Reference

#### 21.3.1 Indefinite Loops

Addr.	Name	Description	
0716B	IDUP	$( \rightarrow )$	
		Pushes interpreter pointer into the stack.	return
071A2	BEGIN	( $\rightarrow$ ) Pushes interpreter pointer into the stack.	return

Addr.	Name	Description
071AB	AGAIN	$(\  ightarrow\ )$
		Sets the interpreter pointer to the topmost
		value in the return stack, without popping it.
071E5	REPEAT	$( \rightarrow )$
		Sets the interpreter pointer to the topmost
		value in the return stack, without popping it.
071C8	UNTIL	( flag $ ightarrow$ )
		If FALSE then AGAIN, otherwise RDROP.
3640F	NOT_UNTIL	( flag $\rightarrow$ )
		NOT then UNTIL.
35B96	#0=UNTIL	$( \# \rightarrow \# )$
		Actually, should be DUP#0=UNTIL.
071EE	WHILE	( flag $ ightarrow$ )
		If TRUE does nothing, otherwise RDROP then
		2SKIP.
36428	NOT_WHILE	( flag $ ightarrow$ )
		NOT then WHILE.
36441	DUP#0<>WHILE	$( \# \rightarrow )$
		Try to guess what it does.

## 21.3.2 Definite Loops

Addr.	Name	Description
073F7	DO	( $\#$ stop $\#$ start $\rightarrow$ )
073C3	ZERO_DO	( $\# stop \rightarrow$ )
364C8	DUP#0_DO	( $\#stop \rightarrow \#stop$ )
073CE	ONE_DO	( $\# stop \  o \ )$
073DB	#1+_ONE_DO	( $\# stop \  o \ )$
364E1	toLEN_DO	$( \ \{\} \ \rightarrow \ \{\} \ )$
		From ONE to #elements.
07334	LOOP	$( \rightarrow )$
073A5	+LOOP	$( \# \rightarrow )$
		Increments index by specified number.
364AF	DROPLOOP	( ob $ ightarrow$ )
36496	SWAPLOOP	( ob1 ob2 $ ightarrow$ ob2 ob1 )
34AAD	SEMILOOP	$( \rightarrow )$
07221	INDEX@	( $\rightarrow$ # )
		Recalls topmost loop counter value.

152 21. Loops

Addr.	Name	Description
3645A	DUPINDEX@	$( ob \rightarrow ob \# )$
3646E	SWAPINDEX@	( ob1 ob2 $ ightarrow$ ob2 ob1 $\#$ )
36482	OVERINDEX@	( ob1 ob2 $ ightarrow$ ob1 ob2 ob1 $\#$ )
367D9	INDEX@#-	$( \ \# \  o \ \#' \ )$
07270	INDEXSTO	$( \ \ \# \  o \ )$
		Stores new topmost loop counter value.
07249	ISTOP@	$( \rightarrow \# )$
		Recalls topmost loop stop value.
07295	ISTOPSTO	$( \ \ \# \  o \ )$
		Stores new topmost loop stop value.
283FC	ISTOP-INDEX	$( \rightarrow \# )$
07258	JINDEX@	$( \rightarrow \# )$
		Recalls second topmost loop counter value.
072AD	JINDEXSTO	$( \# \rightarrow )$
		Stores new second topmost loop counter value.
07264	JSTOP@	$( \rightarrow \# )$
		Recalls second topmost loop stop value.
072C2	JSTOPSTO	$( \ \# \  o \ )$
		Stores new second topmost loop stop value.
3709В	ExitAtLOOP	$(\rightarrow)$
		Does not exit loop immediately. Just stores zero
		as the stop value, so all objects until the next
		LOOP will be evaluated.
		aka: ZEROISTOPSTO

## Chapter 22 Error Handling

When an error occurs in a System RPL program, normally the program is aborted and a message box is popped with the error message. However, sometimes it is desired for the program to trap the error and if possible continue execution, or perhaps show that an error happened in a different way.

Other times, the programs need to *generate* an error. For example, if the user gave invalid input for the program, it should abort with a "Invalid Argument Type" error, instead of risking crashing the machine.

#### 22.1 Trapping Errors

You can intercept the execution of the error handling subsystem, i.e., trap an error generated by your program, using the following structure:

If <suspect objects> and/or <if-error objects> are only a single object, it is not necessary to include them inside a secondary, naturally.

It works like this: if the <suspect objects> generates an error, the execution continues at <if-error objects>. Otherwise, it continues past it.

The action of <if-error objects> is completely flexible. Normally, it will handle the error and then continue or exit the program. The current error number can be recalled with ERROR@, and then your program can do different actions on different kinds of errors. The error messages and numbers can be found in Appendix E.

#### 22.1.1 The Protection Word

Each temporary environment (see Chapter 18), DO/LOOP environment (see Chapter 21) and virtual stack level (see Chapter 23) has a *protection word*. Its purpose is to allow the error handling subsystem to distinguish which environments were created before the error trap, and which were created after. This way, all environments that were created after the error trap was set will be deleted in case of an error. For example, consider the following code:

```
1
    ::
       1LAMBIND
5
       TEN ZERO DO
         ERRSET ::
            . . .
           1LAMBIND
            . . .
10
           FIVE ONE_DO
              <suspect object is here>
           LOOP
           ABND
15
         ERRTRAP ::
           <error handling>
       LOOP
20
       ABND
```

If an error is generated, then the error will be trapped. The inner DO/LOOP and temporary environments will be deleted, thanks to the protection word.

When one of these environments is created, its protection word is set to zero. The word ERRSET increments the protection word of the most recent environment of each of the three kinds. This way, these environments now have a non-zero protection word. (The protection word was initialized to zero when the environment was created.)

The words ERRTRAP and ERRJMP delete these kinds of environments (from the newest to the oldest) until they find one (of each type) with a non-zero protection word. These environments were the ones that already existed before

the setting of the error trap, because they have had their values increased by ERRSET. This way, all environments created after the setting of the trap (which still have the protection word as zero) are deleted. Another effect of ERRTRAP and ERRJMP is that they decrement the protection word of those first environments found with a non-zero protection word, so that the process works correctly if there are several levels of nesting.

#### 22.2 Generating Errors

The error handling subsystem is invoked by the word ERRJMP. If an error trap was set, the error handler will be executed. If none was set, then the default one will be run.

In most cases, when you generate an error, you will let the default error handler deal with it. This default handler does a beep (if this feature is enabled), and displays a description of the error in a message box.

The displayed message depends on two things: the error number, which defines the error message (such as "Bad Argument Type" or "Too Few Arguments") and the last stored command name.

This last stored command name is automatically stored by the CK<n> words described in Chapter 29. As mentioned there, if you are writing a program that is not part of a library, no command name should be stored, because otherwise an ugly name will be shown.

To define the error number, use the word ERRORSTO. It expects a bint as argument: the number of the error. The errors are listed in Appendix E.

There are some words that automate this process, generating some common errors, such as SETTYPEERROR. These words are listed in the reference section below. There are also shortcut words for generating some CAS error messages. These are described in Chapter 52.

Sometimes, however, it is desired to generate an error message that is not in the built-in error list. In order to do that, first you need to store the desired message by means of the command EXITMSGSTO. Then, store #70000 as the error number. Note that there is a built-in bint, called #EXITERR, which contains that number. Now, just call ERRJMP.

The process above can be simplified by using the words DO#EXIT and DO\$EXIT. The first takes a bint as argument, stores that number and calls ERRJMP. The latter is used with strings, it takes a string as argument and does the actions described in the previous chapter. However, both entries also call

AtuserStack, which tells the error handling system not to delete any objects in the stack. So, do not use this word if there are objects in the stack (put by your program) that should be deleted. The automatic deletion of non-user objects in the stack when an error occurs will be described in more detail in section 29.1.

#### 22.3 Reference

#### 22.3.1 General Words

Addr.	Name	Description
26067	ERRBEEP	$(\  ightarrow\ )$
		Beeps.
04CE6	ERROR@	$( \rightarrow \# )$
		Returns current error number.
04D0E	ERRORSTO	$( \# \rightarrow )$
		Stores new error number.
36883	ERROROUT	$( \# \rightarrow )$
		Stores new error number and calls ERRJMP.
04D33	ERRORCLR	$( \rightarrow )$
		Stores zero as new error number.
04ED1	ERRJMP	$( \rightarrow )$
		Invokes error handling sub-system.
04E07	GETEXITMSG	$( \rightarrow \ \ \ \ \ )$
		Gets EXITMSG (user defined error message).
04E37	EXITMSGSTO	$( \ \ \ \ \ \ \ \ \ \ \ )$
		Stores \$ as EXITMSG.
25EAE	DO#EXIT	$(\#\to)$
		Stores new error number, does AtUserStack
		and then ERRJMP.
25EB0	DO%EXIT	( % → )
		Same as above, but takes real number as argu-
		ment.
25EAF	DO\$EXIT	$( \ \ \ \ \ \ \ \ \ \ \ )$
		Stores string as EXITMSG, #70000 as error
		number, does AtUserStack and then ERRJMP.
04EA4	ABORT	$(\rightarrow)$
		Does ERRORCLR and ERRJMP.

Addr.	Name	Description
04E5E	ERRSET	$(\  ightarrow\ )$
		Sets new error trap.
04EB8	ERRTRAP	$( \rightarrow )$
		Error trap marker. If no error happens, still removes all temporary environments created since ERRSET.
04D87	JstGetTHEMESG	( # → \$ )
0 12 0 /		Fetches message from message table. To get a message from a library, use the formula: libnum*#100+msgnum.
		aka: JstGETTHEMSG
04D64	GETTHEMESG	( $\# \to \$$ ) If #70000 then does GETEXITMSG, else does JstGetTHEMESG.
39332	(?GETMSG)	( $\# \to \$msg$ ) ( ob $\to$ ob ) If the argument is a bint, does JstGETTHEMSG to fetch a message. Other arguments are returned unchanged.

# 22.3.2 Error Generating Words

Addr.	Name	Description
04FB6	SETMEMERR	Error 001h
		Generates "Insufficient Memory" error.
05016	SETROMPERR	Error 004h
		Generates "Undefined XLIB Name" error.
04FF2	SETPORTNOTAV	Error 00Ah
		Generates "Port Not Available" error.
26134	SYNTAXERR	Error 106h
		Generates "Invalid Syntax" error.
260C1	NOHALTERR	Error 126h
		Generates "HALT Not Allowed" error.
26116	SETCIRCERR	Error 129h
		Generates "Circular Reference" error.
262E2	SETSTACKERR	Error 201h
		Generates "Too Few Arguments" error.

Addr.	Name	Description
262DD	SETTYPEERR	Error 202h
		Generates "Bad Argument Type" error.
262D8	SETSIZEERR	Error 203h
		Generates "Bad Argument Value" error.
262E7	SETNONEXTERR	Error 204h
		Generates "Undefined Name" error.
2F458	SETIVLERR	Error 304h
		Generates "Undefined Result" error.
2F37B	SetIOPARErr	Error C12h
		Generates "Invalid IOPAR" error.
3721C	Sig?ErrJmp	$( \ \# \  o \ )$
		Calls ERRJMP if the error number is any of {13E
		123 DFF}.
25F10	ederr	$( \rightarrow )$
		Error handler for applications which use
		savefmt1 to save the current display format.
		Calls rstfmt1 and then errors out.

# Chapter 23 The Virtual Stack

The HP49 has a "Virtual Stack" feature. It is a set of commands that can manipulate an RPN Stack: basically, you can save the stack and then restore it.

There exists, in fact, a stack of stacks (a metastack?). The topmost (and in normal conditions, the only) one is the normal RPN stack, in which the user enter objects, and from which commands take and return arguments. This stack will be referred as RPN stack. The set (or, more specifically, the stack) of stacks will be referred as "Virtual Stack", with uppercase initials.

You can push the RPN stack (or part of it), making these pushed objects a level of the "Virtual Stack". A level of the Virtual Stack will be called "virtual stack", with lowercase initials. After pushing the RPN stack, you can manipulate it in any way, and you can at any time restore the contents previously pushed. Or you can push another stack, thus having two stored virtual stacks, in addition to a "new" RPN stack which can be used independently.

Each of these pushed virtual stacks holds a number of objects, and the count of objects. The number of objects is determined when the virtual stack is pushed, and it is not possible to add more objects later. The words that return the virtual stack as a meta return this count, the others do not. When pushing, the words that push the stack as a meta allow you to push only part of the stack; the others push everything in the RPN stack. But you can pop as a meta a stack that was not pushed as one, or push a stack as a meta and pop is not being a meta. The only difference is that the count of elements may or may not be returned.

The Virtual Stack is used in nearly every HP49 application. It is extremely useful (and really fast) when you want to save immediately a complete stack, without using much memory.

It is the Virtual Stack that allows you to enter a full command line in an Input Form and get the results of that command line in the field, for example. Suppose in an InputForm you type DROP. You will get an error, "Too Few Arguments" even if the stack was not empty. Before the HP49 runs the

command, it saves the stack into the Virtual Stack, then run the command. Once the command has been run, it restores the pushed virtual stack above the new one.

The Virtual Stack is located inside a string which is the first object in TEMPOB. It has a similar structure as a Local Variable stack. It is made with blocks, and is protected exactly like local variables. If you trap an error, the virtual stacks created inside the ERRSET and ERRTRAP will be automatically deleted, exactly as are local variable blocks. (See section 22.1.1 for more information.)

For examples of the application of the Virtual Stack, see the Dol list processor in section 19.3 and the HP48 Browser example in section 34.7. Following, there is a list of the commands that deal with the Virtual Stack.

Addr.	Name	Description
25F1E	PushVStack	$( \   obnob1 \ \rightarrow \   obnob1 \ )$
		Virtual Stack:
		$( \rightarrow [obnob1] )$
		Pushes the RPN stack onto the Virtual
		Stack. The RPN stack is unchanged.
25F1F	PushVStack&Clear	( obnob1 $\rightarrow$ )
		Virtual Stack:
		$( \rightarrow [obnob1] )$
		Does PushVStack and then clears the RPN
		stack.
25F1A	PopMetaVStackDROP	$( \rightarrow \text{obnobl})$
		Virtual Stack:
		$([obnob1] \rightarrow )$
		Pops the topmost virtual stack into the RPN
		stack. The previous contents of the RPN
		stack are preserved. (The Meta in the name
		means that a count is returned, but the DROP
		removes it afterwards.)
		•

Addr.	Name	Description
25F1B	PopVStack	( obmob1 → obn'ob1' ) Virtual Stack: ( [obn'ob1'] → ) Pops the topmost virtual stack into the RPN stack. The previous contents of the RPN stack are lost.
25F17	GetMetaVStackDROP	( → obnobl ) Virtual Stack: ( [obnobl] → [obnobl] ) Inserts the objects from the topmost virtual stack into the RPN stack. The Virtual Stack is unchanged. (The Meta in the name means that a count is returned, but it is removed by DROP.)
25F18	GetVStack	( obmobl → obn'obl' ) Virtual Stack: ( [obn'obl'] → [obn'obl'] ) Copies the topmost virtual stack into the RPN stack. The Virtual Stack is not changed, but the current RPN stack is lost.
26265	PushMetaVStack	<ul> <li>( obnobl #n → obnobl #n )</li> <li>Virtual Stack:</li> <li>( → [obnobl] )</li> <li>Pushes #n objects as a new virtual stack.</li> <li>Any other objects in the RPN stack are not pushed. The RPN stack is unchanged.</li> </ul>
25F1D	PushMetaVStack&Drop	( obnobl #n $\rightarrow$ ) Virtual Stack: ( $\rightarrow$ [obnobl] ) Does PushMetaVStack then drops the pushed objects. Any other objects present in the RPN stack are neither pushed nor dropped.

Addr.	Name	Description
25F19	PopMetaVStack	( $\rightarrow$ obnob1 #n )
		Virtual Stack:
		$([obnob1] \rightarrow )$
		Insers the contents of the most recent vir-
		tual stack into the RPN stack, followed by
		the count. The previous contents of the RPN
	_	stack are not lost.
2624C	GetMetaVStack	$(\rightarrow \text{obnob1 } \# \text{n})$
		Virtual Stack:
		$([obnob1] \rightarrow [obnob1])$
		Inserts the objects from the topmost virtual
		stack into the RPN stack, along with the
25F20	DecabitOb a als Citie and	count. The Virtual Stack is unchanged.  ( obnob1 obm'ob1' #m →
Z3FZU	PushVStack&Keep	( ObliObl ObliObl #iii → Obm'obl' #m )
		Virtual Stack:
		$( \rightarrow [obnob1] )$
		Pushes the contents of the RPN stack which
		do not belong to the meta (ie, are "above"
		it) into a new virtual stack, removing these
		elements, but keeping the meta.
25F21	PushVStack&KeepDROP	( obnob1 obm'ob1' $\#m \rightarrow$
		obm'ob1')
		Virtual Stack:
		$( \rightarrow [obnob1] )$
		Does PushVStack&Keep and then DROP.
25F1C	PopVStackAbove	$(obm'obl' \rightarrow obnobl$
		obm'ob1')
		Virtual Stack:
		( [obnob1] → )
		Pops the contents of the topmost virtual
		stack (like PopMetaVStackDROP would
		have done) into the RPN stack, but <i>above</i> the current contents of the RPN stack. This
		undoes PushVStack&Keep (or PushVS-
		tack&KeepDROP).

Addr.	Name	Description
26215	DropVStack	$(\  ightarrow\ )$
		Virtual Stack:
		$([obnob1] \rightarrow )$
		Drops the topmost virtual stack from the Vir-
		tual Stack.
26229	GetElemTopVStack	( #i $ ightarrow$ obi $)$
		Virtual Stack:
		$([obnob1] \rightarrow [obnob1])$
		Returns the ith object from the topmost vir-
		tual stack, counting from the top. "Counting
		from the top" means that object # 0 is the one
		at the highest-numbered level (n), # 1 is the
		one at level n-1, and so on. Note: no checking
		wheter #i is valid.
2626F	PutElemTopVStack	( new_ob #i $ ightarrow$ )
		Virtual Stack:
		$([obnob(n-i)ob1] \rightarrow$
		[obnnew_obob1] )
		Replaces the ith object from the topmost
		virtual stack with new_ob, counting from
		the top. Note: no checking wheter #i is valid.
26224	GetElemBotVStack	(#i → obi )
		Virtual Stack:
		$([obnob1] \rightarrow [obnob1])$
		Returns the ith object from the topmost
		virtual stack, counting from the bottom.
		"Counting from the bottom" means that #0 is
		the object in the lowest numbered level (gen-
		erally thought of as 1), # 1 is at level 2, etc.
26267	D + T]	Note: no checking wheter #i is valid.
2626A	PutElemBotVStack	( new_ob #i → ) Virtual Stack:
		( [obnobiob1] →
		$(\text{obnnew\_obobl}) \rightarrow (\text{obnnew\_obobl})$
		Replaces the ith object from the topmost
		virtual stack with new_ob, counting from
		the bottom. Note: no checking wheter #i is
		valid.
		vanu.

Addr.	Name	Description
26233	GetVStackProtectWord	( → # )
		Hacking stuff: Gets the protection word of
		the last VStack level.
2622E	SetVStackProtectWord	( $\#$ $\rightarrow$ )
		Hacking stuff: Sets the protection word of
		the last VStack level.

# **Chapter 24 Memory Operations**

Word	Stack and Action
CREATE	(ob id $\rightarrow$ )
	Creates a variable with the name id and contents ob. An error
	occours if ob is or contains the current directory ("Directory Re-
	cursion"). This word does not check if there is already a variable
	with name id: even if there is, another one is created.
STO	$(ob id \rightarrow)$
	$( ob lam \rightarrow )$
	In the lam case, the temporary identifier is rebound to ob. An
	error occurs if the lam is unbound. In the id case, STO attempts to
	replace the contents of the variable named id with ob. If a variable
	with that name was not found, a new variable is created.
@	(id $\rightarrow$ ob TRUE)
	$(id \rightarrow FALSE)$
	(lam $\rightarrow$ ob TRUE)
	(lam $\rightarrow$ FALSE)
	Attempts to return the contents stored in the variable or tempo-
	rary identifier. Returns the stored object and TRUE if successful,
	or just FALSE if no variable or lam was found with that name. In
	the case of variables, searching starts in the current directory and
	works upwards through parent directories if necessary.

One problem with STO and @ is that if you give, say, SIN as the argument, the whole body of the function is stored in the variable. For that reason, it is better to use SAFESTO and SAFE@, which work like STO and @, but they automatically convert ROM bodies into XLIB names (SAFESTO) and back again (SAFE@).

Note that the SAFE in these and other entries only means that they do

the conversions described above. With other aspects, there is no safety in these entries.

There are many other words related to memory, which you will find in the list below.

### 24.1 Reference

# 24.1.1 Recalling, Storing and Purging

Addr.	Name	Description
0797B	@	( $id/lam \rightarrow ob T$ )
		( $id/lam \rightarrow F$ )
		Basic recalling function.
35C2C	DUP@	( $id/lam \rightarrow id/lam \ ob \ T$ )
		( $id/lam \rightarrow id/lam F$ )
		Does DUP then @.
35A5B	SAFE@	( $id/lam \rightarrow ob T$ )
		( $id/lam \rightarrow F$ )
		For lams does @. For ids does ?ROMPTR $>$ to the
		ob found.
35A56	DUPSAFE@	( $id/lam \rightarrow id/lam \ ob \ T$ )
		( $id/lam \rightarrow id/lam F$ )
		Does DUP then SAFE@.
25EF7	SAFE@_HERE	( id $ ightarrow$ ob F )
		( id $ ightarrow$ T )
		Same as SAFE@, but works only in the current
		directory.
2F064	Sys@	( ID $ ightarrow$ ob T )
		( ID $ ightarrow$ F )
		Switches temporarily to the HOME directory
		and executes @ there.
2F2A3	XEQRCL	( id $ ightarrow$ ob )
		Same as SAFE@, but errors if variable is not
		found. Also works for lams, but you get the
		wrong error.
2F24E	LISTRCL	( $\{ path id \} \rightarrow ob \}$
		Recalls from specified path.

Addr.	Name	Description
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.
35A29	SAFESTO	( ob id/lam $\rightarrow$ ) For ids, does ?>ROMPTR to the object before storing.
2F380	SysST0	( ob ID $\rightarrow$ ) Switches temporarily to the HOME directory and executes STO there.
25E79	XEQSTOID	( ob id/lam → ) Same as SAFESTO, but will only store in the current directory and will not overwrite a directory. aka: ?STO_HERE
25F0C 3E823	XEQStoKey xSTO>	( ob ID $\rightarrow$ ) ( ob id $\rightarrow$ ) ( ob symb $\rightarrow$ ) Like xSTO, but if the level 1 argument is symbolic, use the first element of it as the variable to write to.
0BD007	^PROMPTSTO1	( $id/lam \rightarrow$ ) Inputs value for a variable and stores it.
085D3	REPLACE	( newob oldob $\rightarrow$ newob ) Replaces oldob (in memory) with newob.
08C27	PURGE	( id $\rightarrow$ ) Purges variable. Does no type check first.
25E78	?PURGE_HERE	( $id \rightarrow$ ) Like PURGE, but only works in current directory.
1D3006	^SAFEPURGE	( $idnt/lam \rightarrow$ ) Purge $idnt/lam$ if it exist.
08696	CREATE	( ob id $\rightarrow$ ) Creates a variable in the current directory. Errors if id is or contains current directory. Assumes id is not a pco.

Addr.	Name	Description
25EC4	DoHere:	$( \rightarrow )$
		Next object in the runstream is evaluated for
		the current directory only.
36A8B	'LAMLNAMESTO	$( ob \rightarrow )$
		STO to LAM LAMLNAME.

### 24.1.2 Directories

Addr.	Name	Description
25EA1	CREATEDIR	( id $\rightarrow$ )
		Creates an empty directory. Calls
		?PURGE_HERE first to delete the original.
08326	LASTRAM-WORD	$( rrp \rightarrow ob T )$
		$( rrp \rightarrow F )$
		Recalls first object in directory.
25EE7	LastNonNull	$( rrp \rightarrow ob T )$
		$( rrp \rightarrow F )$
		Recalls first object in directory (not null
		named).
08376	PREVRAM-WORD	$( ob \rightarrow ob' T )$
		$( ob \rightarrow F )$
		Recalls next object in directory.
25EF2	PrevNonNull	$( ob \rightarrow ob' T )$
		$( ob \rightarrow F )$
		Recalls next object in directory (not null
		named).
082E3	RAM-WORDNAME	( ob $ ightarrow$ id )
		Recalls name of object in current directory.
25F14	XEQPGDIR	$(id \rightarrow )$
		Purges a directory. Checks references, etc.
		first.
2F296	XEQORDER	( {id1 id2} $\rightarrow$ )
		Orders the variables in the directory by mov-
		ing the given variables to the beginning of the
		directory.
25EB9	DOVARS	( $\rightarrow$ {id1 id2} )
		Returns list of variables from current directory.

Addr.	Name	Description
25EB8	DOTVARS%	( % → {} )
		Returns a list of variables in the current direc-
		tory with user type given by the number. In-
		ternal TVARS if a single number was given.
0BD002	^DOTVARS{}	( $\{\# \ \#' \ \ldots\} \rightarrow \{\}$ )
		Returns a list of variables in the current direc-
		tory with user type given by any of the num-
		bers in the list. This is the core of the TVARS
		program.
25EF1	PATHDIR	( $ ightarrow$ {HOME dir1 dir2} )
		Returns current path.
2F265	UPDIR	$( \rightarrow )$
		Goes to parent directory.
08D5A	CONTEXT@	$( \rightarrow \mathtt{rrp} )$
		Recalls current directory.
08D08	CONTEXT!	( rrp $\rightarrow$ )
		Sets new current directory.
08DD4	SYSRRP?	( $ ext{rrp}  ightarrow  ext{flag}$ )
		Is rrp HOME?
08D92	HOMEDIR	$( \rightarrow )$
		Sets HOME as current directory.
		aka: SYSCONTEXT
3712C	SaveVarRes	$( \rightarrow )$
		Binds current and last directories to two null-
		named lams.
37186	RestVarRes	$( \rightarrow )$
		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.

#### 24.1.3 The Hidden Directory

Addr.	Name	Description
3714A	SetHiddenRes	$(\  ightarrow\ )$
		Sets the hidden directory as the current and last
		directories.
370C3	WithHidden	$( \rightarrow ? )$
		Executes next command in hidden directory.
370AF	RclHiddenVar	( id $\rightarrow$ ob T )
		( id $ ightarrow$ F )
		Recalls variable in hidden directory. Same as
		:: WithHidden @ ;
37104	StoHiddenVar	( ob id $ ightarrow$ )
		Stores variable in hidden directory. Same as
		:: WithHidden STO ;
37118	PuHiddenVar	( id $ ightarrow$ )
		Purges variable in hidden directory. Same as
		:: WithHidden PURGE ;

#### 24.1.4 Temporary Memory

The objects in the stack are in a area called "temporary memory". As the name says, it is intended for temporary storage.

When you duplicate an object in the stack, you do not actually create a copy of it: the stack contains only pointers to objects, and only this pointer is duplicated.

When you modify an object, most commands automatically make a new copy of the object in question and modify the copy. In other words, if you enter a string in the stack, press ENTER and edit the string, you have two different strings now. This only happens because a copy of the string was made before editing it. If the copy was not made, the two strings would have been modified, because they were actually the same object.

There are a few commands that do not make a copy of the object before editing it. This means that all copies of the object, in the stack or even stored in memory will be modified at the same time. Sometimes this is desired, sometimes not. These commands are sometimes called "bang type". When this kind of command appears in this book this is noted in their description. When you use these commands, you must be careful not to modify too much objects

simulateneously... You can use the commands  ${\tt TOTEMPOB}$  or  ${\tt CKREF}$  to make another copy of the object: with this, it becomes safe to use this "bang type" commands.

Addr.	Name	Description
06657	TOTEMPOB	( ob $\rightarrow$ ob' )
		Copies object to TEMPOB and returns pointer
		to the new copy.
35C90	TOTEMPSWAP	( ob1 ob2 $\rightarrow$ ob2' ob1 )
		Does TOTEMPOB then SWAP.
25E9F	CKREF	( ob $ ightarrow$ 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.
3700A	SWAPCKREF	( ob1 ob2 $ ightarrow$ ob2 ob1' )
		Does SWAP then CKREF.
06B4E	INTEMNOTREF?	( ob $ ightarrow$ ob flag )
		If the object is in TEMPOB area, is not embed-
		ded in a composite and is not referenced, re-
		turns the object and TRUE, otherwise returns
		the object and FALSE.
01E0E8	~INTEMPOB?	( ob $ ightarrow$ ob flag )

# **Chapter 25 Time and Alarms**

This chapter contains a list of entries related to times, dates and the internal list of alarms.

Addr.	Name	Description
26120	SLOW	$( \rightarrow )$
		15 millisecond delay.
26125	VERYSLOW	$( \rightarrow )$
		300 millisecond delay.
2F37E	SORTASLOW	$( \rightarrow )$
		1.2 second delay (4 x VERYSLOW).
2612A	VERYVERYSLOW	$( \rightarrow )$
		3 second delay.
2F2D4	dowait	( %secs $ ightarrow$ )
		Waits specified number of seconds.
3005E	%>HMS	( % $ ightarrow$ %hms )
		Converts from decimal to H.MMSS format.
30912	%%H>HMS	( %% $\rightarrow$ %%hms )
		Same as %>HMS, but for long reals.
30077	%HMS>	( %hms $\rightarrow$ % )
		Converts from H.MMSS format to decimal.
3008B	%HMS+	( %hms1 %hms2 $\rightarrow$ %hms )
		Adds time in hms format.
300B3	%HMS-	( %hms1 %hms2 $\rightarrow$ %hms )
		Subtracts time in hms format.
2EECF	TOD	( $ ightarrow$ %time )
		Returns current time.

Addr.	Name	Description
2F388	VerifyTOD	( %time $\rightarrow$ %time )
		Checks for validaty of time. Errors if not valid.
2EED0	DATE	$( \rightarrow \text{ %date })$
		Returns current date.
2EED2	DATE+DAYS	( %date %days → %date' )
		Adds specified number of days to date.
2EED1	DDAYS	( %date1 %date2 → %days )
		Returns number of days between two dates.
2EED7	CLKTICKS	$(\rightarrow hxs)$
		Returns tick count.
		aka: SysTime
2EED3	TIMESTR	( %dt %tm $\rightarrow$ "dy dt tm" )
		Returns string representation of time, using
		current format. Example:
		"WED 06/24/98 10:00:45A"
2F329	Date>d\$	( $%date \rightarrow $$ )
		Returns string representation of date, using cur-
		rent format.
2F381	TOD>t\$	( %time $\rightarrow$ \$ )
		Returns string represent the time, using current
		format.
2F1AB	Date>hxs13	( $%date \rightarrow hxs$ )
		Converts date to ticks.
2F003	(Ticks>Date)	$( \text{hxs} \rightarrow \text{%date})$
		Returns date from hxs of internal alarm list for-
		mat.
2F002	(Ticks>TOD)	( hxs $\rightarrow$ %time )
		Returns time from hxs of internal alarm list for-
		mat.
2F004	(Ticks>Rpt)	$( \text{hxs} \rightarrow \text{%rpt} )$
		Converts has in internal alarm list format to
		repetition interval.
		Converts hxs in internal alarm list format to repetition interval.

### 25.1.1 Alarms

The internal alarms list has this format:

```
{ { hxs action } { ... } ... }
```

The length of each hxs is 24 nibbles. The least significant 13 nibbles represent the tick value for the time and date. The next 10 nibbles represent the repeat interval, if any. The most significant nibble represents the status of the alarm (pending, acknowledged, etc.).

Addr.	Name	Description
2F178	ALARMS@	$( \rightarrow \{\} )$
		Returns internal alarms list.
2F37F	STOALM	( %date %time acti %rep $ ightarrow$ % )
		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 re-
		peats every hour, %repetition would be 8192*60*60
		= 29491200. Returns real number representing the
		position of the alarm in the list.
2F0AC	PURGALARM%	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
		Internal DELALARM.
2F314	RCLALARM%	$( n \rightarrow \{ \} )$
		Recalls nth alarm. List is in the format of
		STOALARMLS.
25FA9	ALARM?	( $ ightarrow$ flag )
		Returns TRUE if an alarm is due.

# **Chapter 26 System Functions**

Following, there is a list of functions dealing with the system, such as configuring some aspects of the calculator and turning the calculator off. The functions dealing with user and system flags are also described here.

### 26.1 Reference

### 26.1.1 User and System Flags

Addr.	Name	Description
2614D	SetSysFlag	( ♯ → )
		Sets the system flag with number #.
26044	ClrSysFlag	$( \# \rightarrow )$
		Clears the system flag with number #.
26170	TestSysFlag	( $\#$ $\to$ flag )
		Returns TRUE if system flag is set.
26152	SetUserFlag	$( \# \rightarrow )$
		Set the user flag with number #.
26049	ClrUserFlag	$( \# \rightarrow )$
		Clear the user flag with number #.
26175	TestUserFlag	( $\# \rightarrow \text{flag}$ )
		Returns TRUE if user flag is set.
2F259	RCLSYSF	$( \rightarrow \text{hxs} )$
		Recalls system flags from 1 to 64.
2F25F	(STOSYSF)	$( \text{hxs} \rightarrow )$
		Stores system flags from 1 to 64.
2F23E	DOSTOSYSF	$( \text{hxs} \rightarrow )$
		Stores system flags from 1 to 64, checking for
		changes in LASTARG flag.

Addr.	Name	Description
2F25A	(RCLSYSF2)	( $ ightarrow$ hxs )
		Recalls system flags from 65 to 128.
2F260	(STOSYSF2)	( hxs $\rightarrow$ )
		Stores system flags from 65 to 128.
2F25B	RCLUSERF	( $ ightarrow$ hxs )
		Recalls user flags from 1 to 64.
2F261	(STOUSERF)	$( \text{hxs} \rightarrow )$
		Stores user flags from 1 to 64.
2F25C	(RCLUSERF2)	$( \rightarrow \text{hxs} )$
		Recalls user flags from 65 to 128.
2F262	(STOUSERF2)	$( \text{hxs} \rightarrow )$
		Stores user flags from 65 to 128.
2F3A9	(STOALLF)	( hxs_usr hxs_sys $ ightarrow$ )
		Stores user and system flags from 1 to 64.
		First is user flags, second is system flags.
2F3AA	(STOALLF2)	( hxs_sys1 hxs_usr1 hxs_sys2
		$\texttt{hxs\_usr2} \ \to \ )$
		Expects 4 hxs and stores them as user and
		system flags.
3B76C	(DOSTOALLF2)	$(\hspace{.1cm}\{\hspace{.1cm}\}\hspace{.1cm}\rightarrow\hspace{.1cm})$
		Stores system and user flags. Expects a list
		with two or four hxs. The first two are the
		system and user flags, respectively, from 1 to
		64. The last two, if present, are the system
		and user flags, respectively, from 65 to 128.
25F23	SaveSysFlags	$( \rightarrow )$
		Save system flags in a virtual stack.
25F22	RestoreSysFlags	$( \rightarrow )$
		Restore system flags from virtual stack, pop-
		ping that level.
2ABF0	RunSafeFlags	Run Stream:
		$( ob \rightarrow )$
		Evaluates the next object in the runstream,
		but saves and restores the system flags
		around it. Uses DoRunSafe. This is very use-
		ful.

Addr.	Name	Description
2AB69	RunInApprox	Run Stream:
		$( ob \rightarrow )$
		Eval next object in runstream with system
		flags 20, 21 clear and 22, 105, 102, 120 set.
2AC0E	DoRunSafe	( ob $\rightarrow$ hxs1 hxs2 )
		Evaluate ob and put the system flags as they
		were before the evaluation on the stack. Used
		${f by}$ RunSafeFlags ${f and}$ RunSafeFlagsNoEr-
		ror.
2ABD7	RunSafeFlagsNoError	Run Stream:
		$( ob \rightarrow )$
		:: 'R DoRunSafe 2DROP ;
2EFA5	DOHEX	$( \rightarrow )$
		Switch stack display format of HEX strings to
		hexadecimal.
2EFA8	DODEC	$( \rightarrow )$
		Switch stack display format of HEX strings to
		decimal.
2EFA6	DOBIN	$( \rightarrow )$
		Switch stack display format of HEX strings to
		binary.
2EFA7	DOOCT	$( \rightarrow )$
		Switch stack display of HEX strings to octal.
2EFBF	BASE	$( \rightarrow \# )$
		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.
2605D	DOSTD	$( \rightarrow )$
		Internal version of user word STD.
26053	DOFIX	$( \ \# \rightarrow )$
		Internal version of user word FIX.
26058	DOSCI	$(\ \ \# \  ightarrow\ )$
		Internal version of user word SCI.
2604E	DOENG	$(\ \ \# \  ightarrow\ )$
061-5	C	Internal version of user word ENG.
261A7	savefmt1	$(\rightarrow)$
		Saves the current number format, and
		changes to STD mode.

Addr.	Name	Description
261A2	rstfmt1	$(\  ightarrow\ )$
		Restores the number format saved by savefmt1. Only one set of flags can be saved,
		there is no nesting of these entries.
2FFDB	SETRAD	$( \rightarrow )$
		Set angular mode to RAD.
25EF3	RAD?	( $ ightarrow$ flag )
		Is angular mode RAD?
2FFBD	SETDEG	$( \rightarrow )$
		Set angular mode DEG.
2FFEF	SETGRAD	$( \rightarrow )$
		Set angular mode GRAD.
25EBA	DPRADIX?	( $ ightarrow$ flag )
		Returns TRUE if current radix is ".".

### 26.1.2 General Functions

Addr.	Name	Description
25EB2	DOBEEP	( %freq %dur $\rightarrow$ )
		Beeps. Analog to user function BEEP.
261AC	setbeep	( $\#ms \ \#Hz \rightarrow$ )
		Also beeps.
041A7	TurnOff	$( \rightarrow )$
		Internal OFF.
041ED	DEEPSLEEP	( $ ightarrow$ flag )
		Puts HP into deepsleep mode. Returns TRUE if
		"Invalid Card Data" message.
01118	LowBat?	( $ ightarrow$ flag )
		Returns TRUE if low battery.
0426A	ShowInvRomp	$( \rightarrow )$
		Flashes "Invalid Card Data" message.
2EE5D	?FlashAlert	$( \rightarrow )$
		Displays system warnings.
05F42	GARBAGE	$( \rightarrow )$
		Forces garbage collection.
05F61	MEM	( $\rightarrow$ # )
		Returns amount of free memory in nibbles. Does
		not do garbage collection. (The user word does.)

Addr.	Name	Description
05902	OSIZE	( ob $\rightarrow$ $\#$ )
		Returns object size in nibbles. Forces garbage
		collection.
05944	OCRC	( ob $\rightarrow$ #nib hxs )
		Returns size in nibbles and checksum as hxs.
2F257	OCRC%	( ob $\rightarrow$ hxs %bytes )
		Returns checksum and size in bytes.
2F267	VARSIZE	( id $\rightarrow$ hxs %bytes )
		Returns checksum and size in bytes of specified
		variable.
394C8	INHARDROM?	( ob $ ightarrow$ ob flag )
		Is object address < #80000h?
05AB3	CHANGETYPE	( ob $\#prolog \rightarrow ob'$ )
		Changes prolog of object, does TOTEMPOB.
25F90	>LANGUAGE	( ♯ → )
		Sets the current language for messages. Inter-
		nal version of $x\rightarrow$ LANGUAGE.
25F95	LANGUAGE>	$( \rightarrow \# )$
		Returns the current language for messages. In-
		ternal version of the $xLANGUAGE \rightarrow command$ .
256BE	NOBLINK	$(\rightarrow)$
		Clears the BLINKFLAG, SysNib5.
25E71	?BlinkCursor	$(\rightarrow)$
		Makes the cursor Blink if in App-mode or Edit-
		line.

# **Chapter 27 Serial Communications**

The entries listed here allow the programmer to write programs that communicate with other machines via the HP49G serial interface.

Addr.	Name	Description
2EEBB	SENDLIST	$(\ \{\}\ \rightarrow\ )$
		Internal SEND.
2EEBC	GETNAME	( $\frac{1}{3}$ /id/lam $\rightarrow$ )
		Internal KGET.
2EEBD	DOFINISH	$( \rightarrow )$
		Internal FINISH.
2EEBE	DOPKT	( \$ \$¹ → )
		Internal PKT.
2EEC1	DOBAUD	$(\% \rightarrow)$
		Internal BAUD.
2EEC2	DOPARITY	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
		Internal PARITY.
2EEC3	DOTRANSIO	$( \ \ \ \ \ \ \ \ \ \ \ )$
		Internal TRANSIO.
2EEC4	DOKERRM	$( \rightarrow \ \ \ )$
		Internal KERRM.
2EEC5	DOBUFLEN	$( \rightarrow % 0/1 )$
		Internal BUFLEN.
2EEC6	DOSBRK	$( \rightarrow )$
		Internal SBRK.
2EEC7	DOSRECV	$( \ \ \ \ \ \ \ \ \ \ \ )$
		Internal SRECV.

Addr.	Name	Description
2EEC9	CLOSEUART	$(\  ightarrow\ )$
		Internal CLOSEIO.
2EECB	DOCR	$( \rightarrow )$
		Internal CR.
2EECD	DODELAY	$(\% \rightarrow)$
		Internal DELAY.
2F31A	APNDCRLF	( \$ → \$' )
		Appends carriage return and line feed to string.
2716D	StdIOPAR	$( \rightarrow \{\} )$
		Default IOPAR: { 9600 0 0 0 3 1 }.
2EEBF	GetIOPAR	$( \rightarrow $ %baud % % % % % $)$
		Recalls IOPAR and explodes it into the stack.
2F062	StoIOPAR	$(\ \{\}\  o\ )$
		STO the list of IO parameters in the HOME direc-
		tory in the variable IOPAR.
2F37B	SetIOPARErr	$( \rightarrow )$
		Throws the IOPAR error: "Invalid IOPAR".
2F34F	KVISLF	( \$ → \$' )
		Like KVIS, but insert <cr> in front of each new-</cr>
		line for PC's.
2F34E	KVIS	( \$ → \$' )
		Translate special characters into digraphs for
		ASCII transfer to a PC.
2F34D	KINVISLF	( \$ → \$' )
		Translate digraphs in the string to characters.
		and remove <cr>&gt; from th end of lines.</cr>
2F389	VERSTRING	$(\  ightarrow\ \ \ \ )$
		Returns version string.

# Chapter 28 The HP49 Filer

The HP49 File Manager (Filer for short) allows the programmer to write various applications that deal with files.

Two built-in applications that use the filer are the File Manager and the Font Browser.

### 28.1 Using the Filer

The general entry to call the filer is `FILER\_MANAGERTYPE. It takes three arguments: Filer\_Type, Filer\_Path and Filer\_List.

#### 28.1.1 The Filer\_Type Argument

This argument allows to select the object types that should be displayed. It is a list of the object prologue addresses that will be allowed.

As an example, the HP49 Font Browser only displays fonts, directories and backup objects. So, Filer\_Type is specified as { DOFONT DORRP DOBAK }.

If you want to browse all kinds of objects, then this parameter should be just { ZERO }. Since this is very common, there is an entry that will supply this list as argument to <code>^FILER\_MANAGERTYPE</code>. It is called <code>^FILER\_MANAGER</code>. Using this entry, you only specify the other two arguments.

### 28.1.2 The Filer\_Path Argument

This argument specifies the initial path. It can be:

Value	Meaning
{ }	start in HOME
{ FOO.DIR }	start in HOME/FOO.DIR

# Value Meaning : n: { } start in port n : n: { FOO } start in backup FOO in port n.

#### 28.1.3 The Filer\_List Argument

This argument specifies the menu keys and hard key assignments. It is a list with one element for each menu key. Each menu key is represented by a list with three to five arguments.

The general structure is like this:

Each of the elements in the sublists will be described now.

#### 28.1.3.1 Name\_Item

This specifies what will be displayed in the menu. It can be either a string, a grob or a program which when evaluated returns a string or a grob.

#### 28.1.3.2 Location Item

This allows you to control when the action bound to this menu can be run. This is either a bint or a program that returns a bint when evaluated.

There are five possible values, which are listed in the table below. The "Constant" column lists the name of a constant (defined in the filer.h file in the includes directory of the examples) that you should use when programming.

Value	Constant	Meaning
0	fEverywhere	The action can be run anywhere.
1	fVar	The action can be run only if the user is browsing
		the HOME directory or one of its subdirectories.
2	fNoLib	The action cannot be run if the user is browsing a
		library.
3	fNoBackup	The action cannot be run if the user is browsing a
		backup object in a port.
4	fHomePort	The action can only be run in the root of a port.

### 28.1.3.3 Action\_Item

This will define what will happen when the user presses the softkey corresponding to that menu or the hardkey assignemnt (see section 28.1.3.5).

It is a bint, or a program that returns a bint when evaluated.

It is possible to call a built-in function of the Filer, or define your own. The table below lists the built-in actions available. Again, "Constant" is the name of a constant defined in filer.h.

Value	Constant	Action
0	cBip	Beeps.
1	cInfo	Not implemented in the HP49G.
<b>2</b>	cHexa	Not implemented in the HP49G.
3	cView	Views the object.
4	cArbo	Shows the directory tree.
5	cUp	Moves the highlight up.
6	cMaxUp	Moves the highlight to the first item.
7	cDown	Moves the highlight down.
8	cMaxDown	Moves the highlight to the last item.
9	cSelect	Marks the selected variable.
10	cUpDir	Goes to the parent directory.
11	cDownDir	Visits the highlighted directory.
12	cPreviousMenu	Displays the previous menu page.
13	cNextMenu	Displays the next menu page.
14	CEVAL	Evalutes the highlighted variable.
15	cSwapHeader	Toggles between the two available header lines.
24	cDetails	Toggles between showing information on the
		variables or just their names.
25	cEDIT	Edits the selected variable.
26	cCOPY	Copies the selected variable.

Value	Constant	Action
27	cMOVE	Moves the selected variable.
28	cRCL	Recalls the contents of the select variable.
29	CPURGE	Purges the selected variable.
30	CRENAME	Renames the selected variable.
31	cCRDIR	Creates a directory.
32	CORDER	Reorders the variables in the current directory.
33	cSEND	Sends the select variable using Kermit.
34	CHALT	Suspends the filer temporarily.
35	cEDITB	Edits the select variable in the most appropriate
		editor.
36	crecv	Receives a variable using Kermit.
37	cQUIT	Exits the filer.
38	cPageUp	Scrolls the contents of the filer one screen up.
39	cPageDown	Scrolls the contents of the filer one screen down.
40	cNewObject	Creates a new variable.
41	cSort	Opens a dialog with several options for sorting
		the variables.

To run a custom program, the Action\_Item argument will be in the range 16-23. Each of these seven values specifies what will be in the stack and how your program is going to be called.

The table below lists the calling methods:

Value	Description
16	Recalls only the current path.
	1: Path
17	Recalls the name and the contents of the currently selected object.
	3: Path
	2: Object
	1: Name
18	Equivalent to the above, but deals with multiple selected objects.
	2n+2: Path
	<u> </u>
	5: Object 2
	4: Name 2
	3: Object 1
	2: Name 1
	1: Number of objects (bint)

Value	Description
19	The program is called once for each object. For each object, puts the
	same that calling method #17 puts.
20	Recalls only the name of the current object.
	2: Path
	1: Name
21	Recalls all the selected names.
	n+2: Path
	<b>:</b>
	3: Name 2
	2: Name 1
	1: Number of names (bint)
22	Recalls the current object only in a string of addresses.
	2: Path
	1: String
23	Recalls the selected objects in a string of addresses.
	2: Path
	1: String

Custom calls 22 and 23 will not be described here, as they are not very useful and somewhat more difficult to use.

When the program is called on a library at the root of a port, some special rules apply to the name:

- For calls 17 and 18, the name will be the title of the library. (Like "Emacs 1.09 CD&Pivo".)
- For call 19, the name will be an "L" plus the number of the library, for example, "L1790".
- For calls 20 and 21, the name will the library number as a real number.

#### 28.1.3.4 ExtraProgram\_Item

When using a custom program, this element holds the program to be called.

There are some additional features that can be useful:

• If you launch the program from VAR, your program will start in the current browsed directory.

- A program can only be called on a selected object, except for call 16 which can be run in an empty directory.
- By default, once a program has been run, the screen will be refreshed, the working directory will be parsed again and the current selection will be lost. You can prevent that by leaving FALSE in the stack. Example:

• If you want to force the Filer to exit after the program is run, leave "TakeOver" in the stack. Example:

```
{ "QUIT" fEverywhere BINT16 :: DROP ' TakeOver ; }
which is equivalent to this, which uses the built-in call:
{ "QUIT" fEverywhere cQUIT }
```

#### 28.1.3.5 Key Shortcut

Use this argument to assing a program to a key. This argument is a bint in the form # axx, where a is 0 or 1, meaning without alpha and with, respectively. xx is the key code plus optionally #40 for the Left Shift or #C0 for the right shift.

If you want to assign your program to LeftShift + TOOL, the number will be #049: #09 representing the TOOL key, and #40 for the LeftShift.

**NOTE**: This argument must be the fifth of the list. So, if you are using a built-in call you will have to define the entry as something like this: { "TITLE" fEverywhere cQUIT TakeOver # 12F } This will assing the program to Alpha-ON.

# 28.2 Reference

Addr.	Name	Description
067004	^Filer	$(\  ightarrow\ )$
		Calls the standard filer.
06D004	^FILER_MANAGER	( $\{ ext{path}\}$ $\{ ext{args}\}$ $ o$ )
		Customized Filer, browsing all object types.
06E004	^FILER_MANAGERTYPE	( $\{ ext{types}\}\ \{ ext{path}\}\ \{ ext{args}\}  ightarrow$ )
		$\{args\} = \{ item1 item2 \}$
		<pre>item = {name loc action [prog] [key]} }</pre>
		Customized filer for selected types only.

# Part III

# Input and Output

# **Chapter 29 Checking for Arguments**

In System RPL, it is very important to check if all arguments required by a program are present in the stack, and if they are of a valid type, when that program is directly accessible to the user. In User RPL, you do not have to worry about this: it is done automatically. In System RPL, very few commands do that, so this task is left for the programmer. This may seen at first a disadvantage, but it is in fact an advantage: you just need to check the arguments once, in the beginning of the program. This generates a fast code, differently from User RPL where the arguments are checked in every command.

### 29.1 Number of Arguments

To check for a specific number of arguments, use one of the following commands. They check if there are enough arguments in the stack, and produce a "Too Few Arguments" error if not.

Command	When to use
CKO, CKONOLASTWD	No arguments required
CK1, CK1NOLASTWD	One argument required
CK2, CK2NOLASTWD	Two arguments required
CK3, CK3NOLASTWD	Three arguments required
CK4, CK4NOLASTWD	Four arguments required
CK5, CK5NOLASTWD	Five arguments required

The CK<n> commands save the name of the command in which they are executed, and if an error happens, that name is displayed. (For more details, see Chapter 22.) This means they should only be used in libraries, because if they are not part of a library and there is an error, the error will be shown as something like "XLIB 1364 36 Error:". In programs that are not a part of a library, use CK<n>NOLASTWD, which does not save the name of the command.

Besides checking for the specified number of arguments, these words also "mark" the stack in a way that, if an error happens, the objects that were pushed in the stack by your program can be removed, leaving no junk in the stack. This works by "marking" the stack above the  $n^{th}$  level, where n is the number of required arguments. For example, if your program uses CK2 or CK2NOLASTWD and there are three arguments in the stack, you can image the stack like this:

3:	10.
2:	3.
1:	5.5

This mark is not fixed at this level; instead it moves as elements are pushed or popped. Here is the stack after the program pushes the bint 1:

4:	10.
3:	3.
2:	5.5
1:	¤ 1h

Now, if an error happens in the program, all objects "below" the mark are dropped. This removes all objects pushed by the program, and also the program arguments if they are still in the stack. This is the standard HP49G behavior.

Besides checking for a number of arguments and providing for error recovery, these words also save the arguments as the last arguments, recoverable via the LASTARG User command, provided this is enabled. If an error occours and it is enabled, then the arguments are automatically restored.

For user-acessible programs that take no arguments, you should nevertheless use CKO (or CKONOLASTWD if it is not part of a library), to mark all the objects in the stack as of user ownership and mark the stack for error recovery.

If your program uses a stack-defined number of arguments (like DROPN), use the words CKN or CKNNOLASTWD. These words first check for a real number in level one, and then for the specified number of objects in the stack. The stack is marked at level two, but only the real number is saved in LAST ARG. The real is converted to a bint.

#### 29.2 Argument Type

The words CK&DISPATCH1 and CK&DISPATCH0 are used to allow your program to do different actions based on the types of arguments given to it. They are used like this:

```
1 ...
    CK&DISPATCH1
    #type1 action1
    #type2 action2
5 #type3 action3
    ...
    #type_n action_n
;
```

The type/action pairs are terminated by a SEMI (;). If after the dispatching you want to do some more actions for all argument types, you will need to enclose the whole CK&DISPATCH1 block in another secondary.

This is how CK&DISPATCHO works: it checks if the stack matches the definitions in #type1. If it does, action1 is executed, after which program execution resumes after SEMI. (Each action must be a single object, so if you want to do more than one action, all of them must be included in a secondary, i.e., between :: and :.) If the type definition does not match the stack, then #type2 is checked, and so on. If no match was found, a "Bad Argument Type" error is generated.

Even when your program accepts only one combination of arguments, this command is still useful for checking if the arguments are of the given type.

The difference between CK&DISPATCHO and CK&DISPATCH1 is that the latter, after completing the first pass unsuccessfully, strips all the tags from the arguments, converts zints to reals, and does a second pass. Only after the second pass without a match the "Bad Argument Type" error is generated.

Each type definition is a bint like this: #nnnnn. Each n is an hexadecimal number representing the object in one position of the stack, according to the table below. The first n represents the object in level five, the second in level four, and so on. This way, #00201 represents a complex number in level three, any object in level two and a real number in level one; #000A6 represents a hxs in level two and an id in level one. There are also two-digit object type numbers, ending in F. Each time you use one of these, the number of arguments that can be checked is reduced. For example, #13F4F represents a real

number in level three, an extended real in level 2 and an extended complex in level one.

Dispatch Code	User Type	Object type
0	n/a	Any object
1	0	Real number
2	1	Complex number
3	2	String
4	3, 4 or 29	Array or matrix
5	5	List
6	6	Identifier (global)
7	7	LAM (Temporary identifier)
8	8, 18 or 19	Secondary
9	9	Symbolic
A	n/a	Symbolic class
В	10	Hex string
$\mathbf{C}$	11	Graphics object (GROB)
D	12	Tagged object
${f E}$	13	Unit object
0F	14	Rompointer (XLIB name)
1F	20	Bint
$2\mathrm{F}$	15	Directory
3F	21	Extended real
4F	22	Extended complex
5F	23	Linked array
$6\mathrm{F}$	24	Character
$7\mathrm{F}$	25	Code object
8F	16	Library
9F	17	Backup object
AF	26	Library data
$\operatorname{BF}$	27	Access pointer
$\mathbf{CF}$	30	Font object
DF	27	Minifont object
EF	27	External object 4 (unused)
$\mathbf{FF}$	28	ZINT

There are also the words CK<n>&Dispatch, where <n> is a number from one to five. These words combine CK<n> with CK&DISPATCH1. Because they use CK<n> (and thus save the last command name), they should only be used in library commands.

#### 29.2.1 Examples

By disassembling and studying built-in words, you can learn a lot. Not only about argument checking, but also about many other things.

The TYPE command provides an example of dispatching. Here is its disassembly:

```
1
    ::
       CK1
       ::
         CK&DISPATCHO
 5
         real
                      응0
                      %1
         cmp
                      %2
         str
                      XEQTYPEARRY
         arry
                      %5
         list
10
         id
                      %6
         lam
                      응7
                      TYPESEC (returns 8, 18 or 19)
         seco
         symb
                      응9
         hxs
                      %10
15
                      %11
         grob
         TAGGED
                      %12
         unitob
                      %13
         rompointer %14
         BINT31
                      %20
                            (#)
20
         BINT47
                      %15
         # 3F
                      %21
                            ( 응응 )
                      %22
         # 4F
                            (C%%)
         # 5F
                      %23
                            (LNKARRAY)
           бF
                      %24
                            (CHR)
25
           7F
                      %25
                            (CODE)
         # 8F
                      %16
         # 9F
                      %17
                      %26
                            (Library Data)
         # AF
         # CF
                      % 30 (Font)
30
         # FF
                      % 28 (ZINT)
                      %27
                           (External)
         any
       SWAPDROP
```

In this case, it would have been possible to use CK&DISPATCH1 instead of CK&DISPATCH0, because tagged objects are explicitly listed on the table.

Since the last item on the list is any, type 27 is returned for any other object not listed.

Since TYPE is part of a library, the command CK1&Dispatch could have been used. The reason it did not is to save ROM space. The inner composite is actually the body of the System RPL command XEQTYPE. This way, System RPL programmers can call a function to return the type of the object, without the overhead of checking if *there is* an object, and without no need to duplicate the dispatching mechanism.

Note the object names. They are aliases for built-in bints. See Chapter 2 for a list of built-in bints.

#### 29.3 Reference

Addr.	Name	Description	
262B0	CK0	$(\  ightarrow\ )$	
		Saves current command to LASTCKCMD.	
		Marks stack below level 1 to STACKMARK.	
262B5	CK1	$( ob \rightarrow ob )$	
		Saves current command to LASTCKCMD. Veri-	
		fies 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.	
262BA	CK2	$( ob1 ob2 \rightarrow ob1 ob2 )$	
		Like CK1, but checks for at least two arguments.	
262BF	CK3	( ob1ob3 → ob1ob3 )	
		Like CK1, but checks for at least three argu-	
		ments.	
262C4	CK4	$( ob1ob5 \rightarrow ob1ob5 )$	
		Like CK1, but checks for at least four arguments.	
262C9	CK5	$(ob1ob5 \rightarrow ob1ob5)$	
		Like CK1, but checks for at least five arguments.	
262CE	CKN	( oblobn %n $\rightarrow$ oblobn #n )	
		Checks for a real in level one. Then checks for	
		that number of arguments. Finally, converts the	
		real to a bint.	
26292	CK0NOLASTWD	$(\rightarrow)$	
		Like CKO, but does not save current command.	

29.3. Reference 197

Addr.	Name	Description
26297	CK1NOLASTWD	$( ob \rightarrow ob )$
		Like CK1, but does not save current command.
2629C	CK2NOLASTWD	( ob1 ob2 $\rightarrow$ ob1 ob2 )
		Like CK2, but does not save current command.
262A1	CK3NOLASTWD	$(ob1ob3 \rightarrow ob1ob3)$
		Like CK3, but does not save current command.
262A6	CK4NOLASTWD	$(ob1ob4 \rightarrow ob1ob4)$
		Like CK4, but does not save current command.
262AB	CK5NOLASTWD	$(ob1ob5 \rightarrow ob1ob5)$
		Like CK5, but does not save current command.
25F25	CKNNOLASTWD	( ob1obn %n $\rightarrow$ ob1obn #n )
		Like CKN, but does not save current command.
2631E	CK&DISPATCH0	$( \rightarrow )$
		Dispatches on stack argument.
26328	CK&DISPATCH1	$( \rightarrow )$
		Dispatches on stack arguments, stripping tags
		and converting reals to ZINTS if necessary.
26323	CK&DISPATCH2	$(\rightarrow)$
		Equivalent to CK&DISPATCH1.
26300	CK1&Dispatch	$(\rightarrow)$
		Combines CK1 with CK&DISPATCH1.
26305	CK2&Dispatch	$(\rightarrow)$
		Combines CK2 with CK&DISPATCH1.
2630A	CK3&Dispatch	$(\rightarrow)$
0620=	GTT 4 a D '	Combines CK3 with CK&DISPATCH1.
2630F	CK4&Dispatch	$(\rightarrow)$
06214	GWE o D'	Combines CK4 with CK&DISPATCH1.
26314	CK5&Dispatch	( \rightarrow )
05503		Combines CK5 with CK&DISPATCH1.
25F9A	OLASTOWDOB!	$(\rightarrow)$
		Clears command save by last CK <n> command.</n>
2EF6C	AtUserStack	aka: $0LASTOWDOB!$ , $0LastRomWrd!$
ZEFUC	ALUSELSLACK	· CKONOLASTWD OLASTOWDOB!;
25E9E	CK1NoBlame	·· CRUNOLASIWD ULASIOWDOB: , ( → )
ZOEZE	CVINODIAME	· · · · · · · · · · · · · · · · · · ·
		· · OTHUSIOMOD: CVINOTUSIMO !

Addr.	Name	Description
354CB	'RSAVEWORD	$(\  ightarrow\ )$
		Stores first object in the composite above the ac-
		tual to LASTCKCMD.
		aka: 'RSaveRomWrd
26319	EvalNoCK	$(comp \rightarrow ?)$
		Evaluates composite without saving as
		current command. If first command
		is CK <n>&amp;Dispatch it is replaced by</n>
		CK&DISPATCH1. If first command is CK <n></n>
		it is skipped. Any other first command is also
		skipped!
25F29	(EvalNoCK:)	Run Stream:
		$( ob \rightarrow )$
		EvalNoCK with the next object in the runstream
		as argument.
2A9E9	RunRPN:	Run Stream:
		$( ob \rightarrow )$
		Evaluate the next object in the runstream with
		RPN mode on (i.e. system flag 95 clear). After
		the evaluation, the system flag is restored to its
		old value.

# 29.3.1 Type Checking

Addr.	Name	Description
36B7B	CKREAL	( % → % )
		$(Z \rightarrow %)$
		Checks for real. If a ZINT, convert to real.
		Else SETTYPEERR.
184006	^CK1Z	( $\$/\#/hxs \rightarrow Z$ )
		CHecks for an integer. Converts strings,
		bints or hxs's to zints. Errors for other ob-
		ject types.
185006	^CK2Z	( ob ob' $ ightarrow$ Z Z' )
		Like ^CK1Z, but for two objects.
186006	^CK3Z	( ob ob' ob'' $ ightarrow$ Z Z' Z'' )
		Like ^CK1Z, but for three objects.

29.3. Reference 199

Addr.	Name	Description
3D2B4	CKSYMBTYPE	( → )
		Checks for quoted name (name as symbolic).
2EF07	nmetasyms	( $meta \rightarrow meta$ )
		Checks for meta containing %, C%, unit, id,
		lam or symb.
03C64	TYPE	( ob $ ightarrow$ #prolog )
		Returns address of prolog of object.
3BC43	XEQTYPE	( ob $ ightarrow$ ob %type )
		System version of user word TYPE, but this
		keeps the object.
3511D	TYPEREAL?	( ob $ ightarrow$ flag )
35118	DUPTYPEREAL?	( ob $ ightarrow$ ob flag )
		aka: DTYPEREAL?
3512C	TYPECMP?	( ob $ ightarrow$ flag )
35127	DUPTYPECMP?	( ob $ ightarrow$ ob flag )
3510E	TYPECSTR?	( ob $ ightarrow$ flag )
35109	DUPTYPECSTR?	( ob $ ightarrow$ ob flag )
		aka: DTYPECSTR?
35136	DUPTYPEARRY?	( ob $ ightarrow$ ob flag )
		aka: DTYPEARRY?
3513B	TYPEARRY?	$( ob \rightarrow flag ??? )$
35292	TYPERARRY?	( ob $\rightarrow$ flag )
352AD	TYPECARRY?	( ob $\rightarrow$ flag )
35195	TYPELIST?	$( ob \rightarrow flag )$
35190	DUPTYPELIST?	(ob $\rightarrow$ ob flag)
		aka: DTYPELIST?
3504B	TYPEIDNT?	$( ob \rightarrow flag )$
35046	DUPTYPEIDNT?	( ob $\rightarrow$ ob flag )
350E1	TYPELAM?	$( ob \rightarrow flag )$
350DC	DUPTYPELAM?	( ob $\rightarrow$ ob flag )
194006	^TYPEIDNTLAM?	$( ob \rightarrow flag )$
0-0-4	( )	Tests if ob is ID or lam.
2F0D4	(ILnot?)	( ob $\rightarrow$ ob flag )
25160		Tests if ob is neither an ID nor a LAM.
35168	TYPESYMB?	$( ob \rightarrow flag )$
35163	DUPTYPESYMB?	$( ob \rightarrow ob flag )$
350FF	TYPEHSTR?	$( ob \rightarrow flag )$
350FA	DUPTYPEHSTR?	$( ob \rightarrow ob flag )$
35186	TYPEGROB?	( ob $ ightarrow$ flag )

Addr.	Name	Description
35181	DUPTYPEGROB?	( ob $ ightarrow$ ob flag )
351A4	TYPETAGGED?	( ob $ ightarrow$ flag )
3519F	DUPTYPETAG?	( ob $ ightarrow$ ob flag )
351B3	TYPEEXT?	( ob $ ightarrow$ flag )
		Is ob a unit object?
351AE	DUPTYPEEXT?	( ob $ ightarrow$ ob flag )
		Is ob a unit object?
3514A	TYPEROMP?	( ob $\rightarrow$ flag )
35145	DUPTYPEROMP?	( ob $ ightarrow$ ob flag )
350F0	TYPEBINT?	( ob $ ightarrow$ flag )
350EB	DUPTYPEBINT?	( ob $ ightarrow$ ob flag )
35159	TYPERRP?	( ob $ ightarrow$ flag )
35154	DUPTYPERRP?	( ob $ ightarrow$ ob flag )
3503C	TYPECHAR?	( ob $ ightarrow$ flag )
35037	DUPTYPECHAR?	( ob $ ightarrow$ ob flag )
35177	TYPECOL?	( ob $ ightarrow$ flag )
		Is on a secondary?
35172	DUPTYPECOL?	( ob $ ightarrow$ ob flag )
		Is ob a secondary?
		aka: DTYPECOL?
350D2	TYPEAPLET?	( ob $ ightarrow$ flag )
350CD	DUPTYPEAPLET?	( ob $ ightarrow$ ob flag )
35087	TYPEFLASHPTR?	( ob $ ightarrow$ flag )
35082	DUPTYPEFLASHPTR?	( ob $ ightarrow$ ob flag )
350C3	TYPEFONT?	( ob $ ightarrow$ flag )
350BE	DUPTYPEFONT?	( ob $ ightarrow$ ob flag )
350B4	TYPELNGCMP?	( ob $\rightarrow$ flag )
350AF	DUPTYPELNGCMP?	( ob $ ightarrow$ ob flag )
350A5	TYPELNGREAL?	( ob $ ightarrow$ flag )
350A0	DUPTYPELNGREAL?	( ob $ ightarrow$ ob flag )
35096	TYPEZINT?	( ob $\rightarrow$ flag )
35091	DUPTYPEZINT?	( ob $ ightarrow$ ob flag )
182006	^TYPEZ?	( ob $ ightarrow$ flag )
183006	^DUPTYPEZ?	( ob $ ightarrow$ ob flag )
114007	^TYPEGAUSSINT?	( ob $ ightarrow$ flag )
		Checks if ob is Gaussian integer.
115007	^DTYPEGAUSSINT?	$( ob \rightarrow ob flag )$
		Checks if ob is Gaussian integer.

29.3. Reference 201

Addr.	Name	Description
116007	^DUPTYPEGAUSSINT?	( ob $\rightarrow$ ob flag )
		Checks if ob is Gaussian integer.
187006	^CK1Cext	( ob $ ightarrow$ flag )
		Checks if object is integer or Gaussian inte-
		ger.
181006	^CKALG	$( ob \rightarrow ob )$
		Checks that an object is real/cmplx/unit or
		idnt/lam/symbolic.
25E77	?OKINALG	( ob $ ightarrow$ ob flag )
		Is object allowed in algebraics?
171006	^DTYPFMAT?	( ob $ ightarrow$ ob flag )
		Tests if object is a symbolic matrix.
191006	^IDNTLAM?	( ob $ ightarrow$ ob flag )
		Tests if ob is idnt or lam.
192006	^FLOAT?	( ob $ ightarrow$ ob flag )
		Tests if ob is real or complex.
195006	^REAL?	( ob $ ightarrow$ ob flag )
		Tests if ob is real, zint or hxs.
196006	^TYPEREALZINT?	( ob $ ightarrow$ flag )
		Tests if ob is real, zint or hxs.
193006	^CKSYMREALCMP	$( ob \rightarrow ob )$
		Does "Bad Argument Type" error if ob is not
		a real, complex or symbolics.

# **Chapter 30 Keyboard Control**

There are several ways a System RPL program can get input from the user:

- From the stack;
- Waiting keystrokes from the keyboard;
- Using the internal INPUT;
- Using the internal INFORM;
- Setting up a Parameterized Outer Loop;
- And other methods.

You have already seen how to get input directly from the stack. Using InputLine, ParOuterLoop and input forms will be seen on the following chapters. In this chapter, you will learn how to read keystrokes from the keyboard.

#### 30.1 Key Locations

In User RPL, key representations have the form %rc.p. In System RPL, they are represented by two binary integers. The first, often called, #KeyCode, goes from one (F1 key) to 51 (ENTER key), and represents each key, in order, from left to right and top to bottom. The up arrow is code 10, being considered the fourth key of the second row. The left, down and right arrows have codes 14, 15, 16, respectively, being considered as part of the third row.

The second number, #Plane, represents the modifier states, according to the table below:

#Plane	Modifiers	#Plane	Modifiers
1	None	4	Alpha
<b>2</b>	Left-shift	5	Alpha, left-shift
3	Right-shift	6	Alpha, right-shift

You can convert from one representation to another using:

```
Ck&DecKeyLoc (%rc.p \rightarrow #KeyCode #Plane)
CodePl>%rc.p (#KeyCode #Plane \rightarrow %rc.p)
```

Sometimes, the shift keys are not being treated as modifiers for other keys, but as keys in their own right. Then they have the key codes 40h (leftshift), C0h (right-shift), and 80h (alpha).

On the HP48, only the six key planes listed above existed. The HP49 introduced five more planes, the shift-hold keys. These are shifted keypresses, where the shift key is being held down while the key is pressed. In User RPL, these keys are denoted by adding 0.01 to the %rc.p representation. For example, the keycode 11.21 means holding down left-shift while pressing the F1 key.

In System RPL, shift-hold keys can be encoded in two ways. The first form (which we will call encoding A) leaves the keycode #kc unchanged, and uses new planes #8...#C. The second form (encoding B) uses planes in the range #1...#6 and adds the keycode of the shift key to the keycode #kc.

The following table lists the different encodings for all possible ways to press the F1 key on the HP49G.

		User RPL	A	1	В	3
Plane	Shift Keys	%rc.pl	#kc	#pl	#kc	#pl
1	Unshifted	11.1	1h	1h	1h	1h
2	Left-shift	11.2	1h	2h	1h	2h
3	Right-shift	11.3	1h	3h	1h	3h
4	Alpha	11.4	1h	4h	1h	4h
5	Alpha, left-shift	11.5	1h	5h	1h	5h
6	Alpha, right-shift	11.6	1h	6h	1h	6h
7	Unused					
8	Left-shift-hold	11.21	1h	8h	41h	2h
9	Right-shift-hold	11.31	1h	9h	C1h	3h
10	Alpha-hold	11.41	1h	Ah	81h	4h
11	Alpha, left-shift-hold	11.51	1h	Bh	41h	5h
12	Alpha, right-shift-hold	11.61	1h	Ch	C1h	6h

Most, but not all, System RPL entries dealing with keys can handle shift-hold key presses. The reference section has information about this issue for each relevant entry. The System RPL entries which expect #kc and #pl as arguments (like CodePl>%rc or Key>StdKeyOb) accept both forms of encoding (A and B). Entries which return #kc or #kc and #pl (like Ck&DecKeyLoc and GETTOUCH) all use encoding B. Encoding A seems to be more convenient for dispatching. In order to convert encoding B into encoding A, you can use

```
:: SWAP 64 #/ ROTSWAP #0=?SKIP #6+ ;
```

#### 30.2 Waiting for a Key

A convenient entry used to wait for a key is WaitForKey. This command puts the HP49 in a low-power state and waits until a key is pressed. It then returns the key code to level two and the plane to level one. There are other words, listed below, which are used in other circumstances.

Unfortunately, WaitForKey does not deal with the shift-hold keys. We therefore list below a short program which behaves just like WaitForKey but returns the extended keycode (encoding B). The program contains a code object accessing the key buffer, which we list without explanation. You will find this program in the keyboard directory in the examples directory.

```
1
    ::
                           (normal WaitForKey)
      WaitForKey
      CODE
                           (get extended keycode)
         GOSBVL =SAVPTR
5
         D0 =
                 047DF
         A=DATO A
         D0=A
         A=0
         A=DAT0 1
10
         A=A-1
         A=A+A
                Α
         CD0EX
         C=C+A
                Α
         CD0EX
15
         D0 = D0 + 2
         A=DAT0 B
         GOVLNG = PUSH#ALOOP
```

30.3. Reference 205

If you would like the program to return encoding A instead of encoding B, just replace ROTDROPSWAP with:

BINT63 #>case #6+

#### 30.3 Reference

## 30.3.1 Converting Keycodes

Addr.	Name	Description
25EA7	Ck&DecKeyLoc	( %rc.p → #kc #p )
		Converts from user key representation format
		to system. Does handle shift-hold keys.
25EA9	CodePl>%rc.p	( $\#kc \#p \rightarrow \ensuremath{\mbox{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath{\ensuremath}\ensuremath$
		Converts from system key representation for-
		mat to user. Does handle shift-hold keys.
25EDC	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
25EEA	ModifierKey?	( $\#kc \#pl \rightarrow flag$ )
		Is the key any of the three modifiers right-shift,
		left-shift, or alpha?

## 30.3.2 Waiting for Keys

Addr.	Name	Description
261CA	FLUSHKEYS	$( \rightarrow )$
		Flushes the key buffer.
		aka: FLUSH

Addr.	Name	Description
04708	CHECKKEY	<ul> <li>( → #kc T )</li> <li>( → F )</li> <li>Returns next key in the key buffer (if there is one), but does not pop it. Does handle shift-hold keys.</li> </ul>
04714	GETTOUCH	Reys.  ( $\rightarrow$ #kc T )  ( $\rightarrow$ F )  Pops next key from key buffer (if there is one).  Does handle shift-hold keys.
25ED6	GETKEY	<ul> <li>( → #kc flag )</li> <li>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 shifthold keys.</li> </ul>
25ED7	GETKEY*	<ul> <li>( → #kc T )</li> <li>( → F F )</li> <li>( → {Alrmlist} T F )</li> <li>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.</li> </ul>
25ED9	GetKey0b	( → ob ) Wait for a single key and return the object associated with this key. Does handle shift-hold keys.
25EC5	DoKey0b	( $ob \rightarrow$ ) 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.
25EE3	KEYINBUFFER?	( $\rightarrow$ flag ) Returns TRUE if there is at least a key in the key buffer.
25F0B	WaitForKey	( $\rightarrow$ #kc #flag ) Returns next full key press. Does <i>not</i> handle shift-hold keys.

30.3. Reference 207

Addr.	Name	Description
2F268	Wait/GetKey	( $\$ \to ?$ ) Internal WAIT command. Does $not$ handle shift-
		hold keys.

# 30.3.3 The ATTN Flag

Addr.	Name	Description
25FAE	ATTN?	( $ ightarrow$ flag )
		Returns TRUE if CANCEL has been pressed.
25E70	?ATTNQUIT	$( \rightarrow )$
		If CANCEL has been pressed, ABORTs program.
		aka: ?ATTN_QUIT
25E9D	CK0ATTNABORT	$( \rightarrow )$
		Executed by the UserRPL program delimiters
		x<< and x>> and by xUNTIL. Mainly just
		?ATTNQUIT.
25EED	NoAttn?Semi	$( \rightarrow )$
		If CANCEL has been not pressed, drops the rest
		of the stream.
05040	ATTNFLG@	$( \rightarrow \# )$
		Recalls CANCEL key counter.
05068	ATTNFLGCLR	$( \rightarrow )$
		Clears CANCEL key counter. Does not affect the
		key buffer.

## **30.3.4 Bad Keys**

Addr.	Name	Description
25EBF	DoBadKey	$( \rightarrow )$
		Beeps.
25ECD	DropBadKey	$( ob \rightarrow )$
		Beeps.
25E6E	2DropBadKey	( ob ob' $ ightarrow$ )
		Beeps.

#### 30.3.5 User Keys

If no keys are assigned, the internal key assignments list is an empty list. If there is one or more assignments, the list contains 51 sublists, each one representing one key. Each sublist is either empty, if that key has no assignments; or contains twelve elements: each representing the assignment of one plane. The planes are given in the table in section 30.1. For planes with no assignment, an empty list must be given. The seventh list is always empty.

Addr.	Name	Description
25F09	UserKeys?	$(  ightarrow  ext{flag})$
		Does BINT62 TestSysFlag.
25967	GetUserKeys	$( \rightarrow \{\} )$
		Returns user keys list (internal format).
2F3B3	(AsnKey)	( ob $\#kc \#p \rightarrow$ )
		Assigns an object to a key, specified in system
		format.
25621	(NonUsrKeyOK?)	( $\rightarrow$ flag )
		Returns TRUE if the keys not defined do their
		normal actions.
25617	(SetNUsrKeyOK)	$( \rightarrow )$
		Keys not defined do their normal actions.
2561C	(ClrNUsrKeyOK)	$( \rightarrow )$
		Keys not defined just beep when pressed.
25EE5	Key>StdKeyOb	( $\#kc \#pl \rightarrow ob$ )
		Recalls the standard assignment of the key.
		This is the assignment which is active when
		USER mode is of.
25EE6	Key>U/SKeyOb	( $\#kc \#pl \rightarrow ob$ )
		If user mode is on, recalls the user object as-
		signed to a key. If user mode is off, recalls the
		standard assignment instead.
255006	^KEYEVAL	( % → ? )
		Keystroke evaluation. If % is negative, the
		standard key is always evaluated.

# **Chapter 31 Using InputLine**

The command InputLine is the system equivalent to the user command INPUT. Its use is similar, and does a similar thing:

- Displays a prompt in the top of the screen;
- Starts the keyboard entry modes;
- Initializes the edit line;
- Accepts input until ENTER is pressed;
- Parses, evaluates, or just returns the user input;
- Returns TRUE if the environment was exited by ENTER or FALSE if it was aborted by ON/CANCEL.

The stack must contain the following parameters:

Name	Description
\$Prompt	The prompt to be displayed during input.
EditLine	The initial edit line.
CursorPos	The initial cursor position. You can either specify the character
#Ins/Rep	position, in absolute terms, or as a two-element list with the row and column. In both cases, a #0 represents the end of edit line, row or column. All numbers should be specified as bints, naturally.  The initial insert/replace mode of the cursor:  • #0 current mode  • #1 insert mode  • #2 replace mode

Name	Description
#Entry	The initial entry mode:
	<ul> <li>#0 current entry mode plus program entry mode</li> </ul>
	• #1 only program entry mode
	<ul> <li>#2 program and algebraic entry modes</li> </ul>
#Alphalock	The initial alpha mode:
	• #0 current mode
	• #1 alpha enabled
	• #2 alpha disabled
ILMenu	The initial menu, in the format specified below. Normally,
	specified as FALSE, which means that the menu should not be changed.
#ILMenu	The initial menu row number (normally BINT1, to show the
	first page).
AttnAbort?	A flag:
	<ul> <li>TRUE CANCEL aborts the input</li> </ul>
	<ul> <li>FALSE CANCEL just clears the edit line</li> </ul>
#Parse	How to process the edit line:
	<ul> <li>#0 return edit line as a string (unevaluated)</li> </ul>
	<ul> <li>#1 return edit line as a string and a parsed object</li> </ul>
	• #2 parse and evaluate edit line

If AttnAbort? is TRUE and the user presses CANCEL while InputLine is active, the edition is aborted. If it is FALSE, CANCEL just clears the edit line. If it was already empty, then it aborts InputLine, returning FALSE.

Depending on the value of #Parse, different values are returned, acording to the table:

#Pa	rse Stack	Description
#0	\$Editline TRUE	Edit line only (unevaluated)
#1	\$Editline obs TRU	JE Edit line and parsed object(s)
#2	ob1 obn TRUE	E Resulting object(s)
	FALSE	CANCEL pressed to abort

#### 31.1 Menu Key Assignments

Any application can specify an initial menu via the ILMenu parameter. This menu will be displayed when the InputLine starts. All menu keys can have assignments to the unshifted, left-shifted and right-shifted planes. When the loop exits, the previous menu is restored intact.

The ILMenu parameter is a list (or, in rare cases, a program returning a list), in the format described in section 37.1. You can also supply just FALSE as this parameter, if you do not want the current menu to be changed.

Note that the actions must start with the word TakeOver to flag that they should be run with the command line active.

#### 31.2 An Example

Here is an example of InputLine, which prompts for your name, and if the edition was not aborted, displays it.

```
1
   ::
      $ "Your name:"
                          (prompt)
      NULL$
                          (initial edit line)
      #ZERO#ONE
                          (cursor at end, insert mode)
5
      ONEONE
                          (prog mode, alpha enabled)
      NULL{}
                          (no menu)
      ONE
                          (menu row)
      FALSE
                          (CANCEL clears)
      ZERO
                          (returns string)
10
      InputLine
      NOT?SEMI
                          (exit if FALSE)
      $ "Your name is "
                          (concatenate string & name)
      SWAP&$
      CLEARLCD
                          (clear display)
15
      DISPROW1
                          (display string on 1st line)
                          (freeze display)
      SetDAsTemp
    ;
```

# 31.3 Reference

Addr.	Name	Description
2EF5F	InputLine	( args $\rightarrow$ \$ T )
		( args $\rightarrow$ \$ ob1obn T )
		( args $\rightarrow$ oblobn T )
		( args $ ightarrow$ F )
		args = \$pr \$line #pos
		#I/R #I/A #alph
		menu #row attn #parse
2F154	(input\$)	( $\$1$ $\$2$ $\rightarrow$ $\$3$ )
		This is what the User command INPUT does if level
		1 is a string.
2F155	$(input{})$	( $$1 {} {} {} {} \rightarrow {} {} {} {} {} {} {} {} {} {} {} {} {} $
		This is what the User command INPUT does if level
		1 is a list.

# **Chapter 32 The Parameterized Outer Loop**

The Parameterized Outer Loop is a System RPL structure that allows you to create a complete application, which receives keystrokes and does different actions based on the key that was pressed. This is repeated as many times as necessary, until an exit condition happens. Most of the time, there is a key that stops the loop, like CANCEL or DROP. Generally, it is used with programs that work with the display. Complex uses of the POL include input forms (Chapter 35) and the browser (Chapters 33 and 34). Note that POLs are a very general construct and for that reason they require elaborate arguments. Simple applications can sometimes be implemented more easily and compactly with a loop around WaitForkey (section 30.2) and direct display handling.

To set up a parameterized outer loop, nine parameters are necessary:

Parameter name	Description
AppDisplay	This object is evaluated before each key evaluation.
	It should handle display updating not handled by the
	keys themselves, and should also perform special han-
	dling of errors.
AppKeys	The hard key assignments, in the format described be-
	low.
NonAppKeyOK?	A flag: if TRUE, then the hard keys not assigned per-
	form their normal actions. Otherwise, they just beep.
DoStdKeys?	A flag: if TRUE, then standard key definitions are used
	for non-application keys instead of default key process-
	ing.
AppMenu	Either the menu specification, in the format described
	in section 37.1, or FALSE to leave the current menu un-
	changed.
#AppMenuPage	The initial menu page. Normally BINT1 to show the
	first page.

Parameter name	Description
SuspendOK?	A flag: if TRUE, any user command that would create
	a suspended environment and restart the system outer
	loop will instead generate an error.
ExitCond	This object is evaluated before each display update and
	key evaluation. If the result is TRUE, the loop is exited.
AppError	The error-handling object to be evaluated in an error occurs during key evaluation.

After setting up the arguments, call ParOuterLoop. This word does not generate any results itself, but any of the key assignments can return results to the stack or any other form desired.

## 32.1 Parameterized Outer Loop Words

The parameterized outer loop is formed by calls (with proper error handling) to the following words. None of them return anything, and the only one that takes arguments is POLSetUI: the same nine required by ParOuterLoop.

Word	Action
POLSaveUI	Saves the current user interface in a temporary environ-
	ment.
POLSetUI	Sets the current user interface, according to the parame-
	ters given.
POLKeyUI	Displays, reads and evaluates keys. Handles errors,
	and exits according to the user interface specified by
	POLSetUI.
POLRestoreUI	Restores the user interface saved by POLSaveUI and
	abandons the temporary environment.
POLResUI&Err	Restores the user interface and errors. This is used when
	there is an error not handled within the parameterized
	outer loop.

The word ParOuterLoop decompiles to:

```
5 POLKeyUI (handle keypresses);

ERRTRAP

POLResUI&Err (if an error happened, restore)

(the saved interface and error)

10 POLRestoreUI (restore saved user interface)
```

If you use the words above instead of ParOuterLoop, you must provide the same level of error protection as the code above.

One note: the parameterized outer loop creates a temporary environment when it saves its current user interface, and it abandons it when it restores a saved user interface. This means that you cannot use words that operate on the topmost temporary environment, like 1GETLAM within the loop, unless the variable was created after calling POLSaveUI, and it is abandoned before calling POLRestoreUI. For temporary environments created before calling POLSaveUI, named temporary variables should be used.

#### 32.2 The Display

In the parameterized outer loop, the user is responsible for setting up the display and updating it; there is no default display.

The display can be updated in two ways: with the parameter "AppDisplay" or with key assignments. For example, when the user presses a key to move the cursor, the key assignment can either pass information to "AppDisplay" (often implicitly), so that it handles the screen updating, or the key assignment object can handle the display itself. Which method is more efficient depends on the situation. In our example below, AppKeys just sets the position of the grob in lams, and AppDisplay draws the grob.

#### 32.3 Error Handling

If an error occurs during the key processing, AppError is executed. This object is responsible for processing any errors generated while the parameterized outer loop is running. AppError should determine the specific error and act accordingly. Or you can just specify ERRJMP as AppError, which means your application does not handle any errors.

#### 32.4 Hard Key Assignments

In the parameterized outer loop, any key in any of the six basic planes (see section 30.1) can be assigned a new function. The parameter AppKeys specifies which keys to assign and their actions.

If a key is not assigned by the application, and the NonAppKeyOK? parameter is TRUE, the standard key definition is executed if the DoStdKeys? parameter is TRUE, or, if available, the USER key assignment, if it is FALSE. If NonAppKeyOK? is FALSE, a warning beep is produced, and nothing else is done.

Most of the time, NonAppKeysOK? should be set to FALSE.

The AppKeys parameter is a secondary, which must take as argument the keycode and plane, and return either the desired key definition and TRUE, or FALSE if the application does not handle it. Specifically, the stack diagram is as follows:

```
( \#KeyCode \ \#Plane \rightarrow KeyDef \ TRUE ) ( \#KeyCode \ \#Plane \rightarrow FALSE )
```

The suggested form for the key assignments is:

```
1 BINT1 #=casedrop :: (process unshifted plane) ;
   BINT2 #=casedrop :: (process left-shifted plane) ;
   ...
2DROPFALSE
```

And each plane handler normally has the form

The word ?CaseKeyDef is very handy in this case, because it is equivalent to #=casedrop :: ' <keydef> TRUE :. Using this word, the code becomes shorter, and the definitions become more legible. ?CaseKeyDef is used in the form:

```
... #KeyCode #TestKeyCode ?CaseKeyDef <keydef> ...
```

If #TestKeyCode equals #KeyCode, ?CaseKeyDef drops both of them, pushes <KeyDef> and TRUE to the stack, and exits the secondary. Otherwise, it drops only #TestKeyCode, skips <KeyDef> and continues.

If you want to handle shift-hold keys, you can do so. The extended key-code (encoding B, see section 30.1) is provided to the AppKeys program on stack levels 5 and 6 All you need to do is to start AppKeys with the snippet

4DROP 2DUP 2DUP

and than dispatch normally.

#### 32.5 Menu Key Assignments

You can specify a menu to be displayed when the parameterized outer loop starts. The format of the AppMenu parameter is essentially the same of the ILMenu parameter of InputLine, described in section 37.1.

The difference is that TakeOver is not necessary in this case, since the input line is not active.

Also, since hard key assignments have priority over menu key assignments, you should put this code in the AppKeys parameter, in each plane definition:

DUP#<7 casedrpfls

This will push FALSE when a key whose code is less than seven (that is, one of the softkeys) is pressed. The FALSE will force the standard assignment to be run, and this assignment runs the action defined by the AppMenu parameter.

For that to work, the NonAppKeysOK? parameter must be TRUE, so that the menu keys work normally, that is, doing the actions specified by the AppMenu parameter.

#### 32.6 Preventing Suspended Environments

Your application may require the evaluation of arbitrary commands and user arguments, but it might not want the current environment to be suspended by HALT or PROMPT commands. The parameter SuspendOK?, when FALSE, will cancel these and any other commands that would suspend the environment and generate a "HALT Not Allowed" error, which AppError can handle. If the parameter is TRUE, the application must be prepared to handle the consequences. "The dangers here are many and severe", as it is written in RPLMAN.DOC.

Almost all applications should set FALSE as the SuspendOK? parameter.

#### 32.7 The Exit Condition

The parameter ExitCond is an object that is evaluated before each key evaluation. If it evaluates to TRUE, the loop is exited, otherwise it continues. You could define, for example, ExitCond as 'LAM exit. When the "quit" key is pressed, you just have to use TRUE 'LAM exit STO and the loop will be exited. Naturally you must create the lam and initialize it with FALSE before.

#### 32.8 An Example

The following program is an example of an application that uses a parameterized outer loop to create an environment where the user may move a little graphic over the screen. You can use the arrow keys to move, or the menu keys. In both cases, if you press left-shift before, the graphic moves ten steps instead of one. There is code to assure that the graphic does not go off the screen boundaries.

Figure 32.1 below displays this program running.



Figure 32.1: The POL example

```
1 ::
    * Defines names for used keys. Makes things easier and
    * more readable
    DEFINE kpNoShift BINT1
5 DEFINE kpLeftShift BINT2
    DEFINE kcUpArrow BINT10
    DEFINE kcLeftArrow BINT14
    DEFINE kcDownArrow BINT15
```

```
DEFINE kcRightArrow BINT16
10
      DEFINE kcLeftShift BINT37
      DEFINE kcOn
    * Requires no arguments
      CK0NOLASTWD
15
    * Prepare display
      RECLAIMDISP
                     (clear and resize display)
                     (temporarily disable clock)
      ClrDAlIsStat
20
    * Smiling face grob. The below must be in one line only.
      GROB 7C 310003100008F000060300810C004000104000102000202
    4012010004010004010004010004011044021042026032048F0104000
    10810C0006030008F000
      FIFTYSIX
                     (initial x coordinate for box)
25
      EIGHTEEN
                     (initial y coordinate for box)
      FALSE
                     (initial exit condition)
        LAM MrSmile
        LAM x
30
        LAM y
        LAM exit?
      } BIND
                     (binds local variables)
    * The following composite is the display update object.
35
    * It clears the screen and draws the smiling face grob.
        CLEARVDISP
                          (clear display)
        LAM MrSmile
                          (recall smiling face grob)
        HARDBUFF
                          (recall current display)
40
        LAM x LAM y
                          (smile coordinates)
        GROB!
                          (REPL)
        DispMenu.1
                          (display menu)
45
    * The following composite is the key action handler.
        kpNoShift #=casedrop ::
          DUP#<7 casedrpfls (enable softkeys)</pre>
          kcUpArrow ?CaseKeyDef
50
            ::
              LAM y DUP
              BINT1 #<ITE
                 :: DROP ERRBEEP ;
```

```
:: #1- ' LAM y STO ;
55
             ;
          kcDownArrow ?CaseKeyDef
             ::
              LAM y DUP
              BINT36 #>ITE
60
                :: DROP ERRBEEP ;
               :: #1+ ' LAM y STO ;
             ;
          kcLeftArrow ?CaseKeyDef
             ::
65
              LAM x DUP
              BINT1 #<ITE
                :: DROP ERRBEEP ;
               :: #1- ' LAM x STO ;
70
          kcRightArrow ?CaseKeyDef
             ::
              LAM x DUP
              BINT111 #>ITE
                :: DROP ERRBEEP ;
75
               :: #1+ ' LAM x STO ;
             ;
          kcOn ?CaseKeyDef
             :: TRUE ' LAM exit? STO ;
          kcLeftShift #=casedrpfls
80
          DROP 'DoBadKeyT
        kpLeftShift #=casedrop ::
          DUP#<7 casedrpfls (enable softkeys)</pre>
          kcUpArrow ?CaseKeyDef
85
             ::
              LAM y DUP
              BINT10 #<ITE
                :: DROPZERO ERRBEEP ;
                :: BINT10 #- ;
90
               ' LAM y STO
             ;
          kcDownArrow ?CaseKeyDef
             ::
              LAM y DUP
95
              BINT27 #>ITE
                 :: DROP BINT27 ERRBEEP ;
                #10+
               ' LAM y STO
```

```
;
100
           kcLeftArrow ?CaseKeyDef
             ::
               LAM x DUP
               BINT10 #<ITE
                :: DROPZERO ERRBEEP ;
105
               :: BINT10 #- ;
               ' LAM x STO
             ;
           kcRightArrow ?CaseKeyDef
             ::
110
               LAM x DUP
               BINT102 #>ITE
               :: DROP BINT112 ERRBEEP ;
                #10+
               ' LAM x STO
115
           kcLeftShift #=casedrpfls
           DROP 'DoBadKeyT
         2DROP 'DoBadKeyT
120
     * Key definitions
       TrueTrue
125
     * Menu specification
       } "qU" }
                  ::
                    LAM y DUP
                    BINT1 #<ITE
130
                      :: DROP ERRBEEP ;
                     :: #1- ' LAM y STO ;
                    LAM y DUP
135
                    BINT10 #<ITE
                      :: DROPZERO ERRBEEP ;
                     :: BINT10 #- ;
                    ' LAM y STO
140
           "Down" { ::
                      LAM y DUP
```

```
BINT36 #>ITE
145
                         :: DROP ERRBEEP ;
                        :: #1+ ' LAM y STO ;
                     ;
                     ::
                       LAM y DUP
150
                       BINT37 #>ITE
                        :: DROP BINT37 ERRBEEP ;
                        #10+
                      ' LAM y STO
155
           "Left" { ::
                       LAM x DUP
                       BINT1 #<ITE
160
                         :: DROP ERRBEEP ;
                        :: #1- ' LAM x STO ;
                     ::
                       LAM x DUP
165
                       BINT10 #<ITE
                        :: DROPZERO ERRBEEP ;
                       :: BINT10 #- ;
                       ' LAM x STO
170
           "Right" { ::
                        LAM x DUP
                        BINT111 #>ITE
175
                         :: DROP ERRBEEP ;
                         :: #1+ ' LAM x STO ;
                      ;
                      ::
                        LAM x DUP
180
                        BINT102 #>ITE
                         :: DROP BINT112 ERRBEEP ;
                         #10+
                        ' LAM x STO
185
         NullMenuKey
         { "Quit" :: TRUE ' LAM exit? STO ; }
```

32.9. Reference 223

```
}
190 ONEFALSE (first menu row, no suspended envs)
' LAM exit? (exit condition)
'ERRJMP (error handler)
ParOuterLoop (run the par outer loop)
RECLAIMDISP (resize and clear display)
195 ClrDAsOK (redraw display)
:
```

## 32.9 Reference

Addr.	Name	Description
2B475	ParOuterLoop	( Disp Keys NonAppKeys? DoStdKeys? menu #row suspendOK? ExitCond AppErr
2B4AC	POLSaveUI	<pre>→ ) ( Disp Keys NonAppKeys? DoStdKeys? menu #row suspendOK? ExitCond AppErr → )</pre>
2B542	POLSetUI	Saves current UI to LAMSavedUI. <see>ParOuterLoop Sets new UI, same arguments as to ParOuterLoop.</see>
2B628	POLKeyUI	$(\rightarrow)$ Displays, reads and evaluates keys according to set UI.
2B6CD	POLRestoreUI	( $\rightarrow$ ) Restores saved UI from LAMSavedUI.
2B6B4	POLResUI&Err	( $\rightarrow$ ) Restores saved UI and executes ERRJMP.
29F25	AppDisplay!	$( ob \rightarrow )$
29F35	AppDisplay@	$( \rightarrow )$
29F55	AppKeys!	$( ob \rightarrow )$
29F75	AppKeys0	???
2A055	AppExitCond!	$( ob \rightarrow )$
2A065	AppExitCond@	
2A145	AppError!	
2A158	AppError@	$( \rightarrow ob )$

Addr.	Name	Description
25690	AppMode?	$(  ightarrow  ext{flag})$
		Is currently a POL active?
25695	SetAppMode	$( \rightarrow )$
2569A	ClrAppMode	$( \rightarrow )$
2564D	SetNAppKeyOK	$( \rightarrow )$
2565A	DoStdKeys?	( $ ightarrow$ flag )
2565F	SetDoStdKeys	$( \rightarrow )$
25F04	SuspendOK?	( $ ightarrow$ flag )
		Does the current user interface allow suspen-
		sion?
27E72	nohalt	( $ ightarrow$ ob )
		:: LAM 'nohalt ;
25671	SetAppSuspOK	$( \rightarrow )$
25676	ClrAppSuspOK	$( \rightarrow )$
	25690 25695 2569A 2564D 2565A 2565F 25F04 27E72	25690 AppMode?  25695 SetAppMode 2569A ClrAppMode 2564D SetNAppKeyOK 2565A DoStdKeys? 2565F SetDoStdKeys 25F04 SuspendOK?  27E72 nohalt 25671 SetAppSuspOK

# Chapter 33 Using the HP49 Browser

The browser is the engine behind the selection boxes created by the User RPL command CHOOSE. However, it can do much more than what that command does.

There are two browser engines in the HP49G calculator: the old one, which was present since the HP48G series, and a new one, only present in the HP49G model. This chapter will describe the new engine, which is easier to use. It has some features the old one does not have, but the old one also has some important features that this one does not, such as full screen mode and selecting multiple items. The next chapter will describe the old engine.

There are several flashpointers which can be use to access the browser engine. These flashpointers are not officially supported, but are very likely stable.

The main difference to User RPL CHOOSE command is that you can specify a message handler, which can be used to provide a custom the menu, to handle key presses and some other things.

The main entry is FPTR 2 72 (^Choose3). It has the following stack diagram:

```
(meta $title #initial ::message \rightarrow ob TRUE) or (meta $title #initial ::message \rightarrow FALSE) depending on whether the user selects something or cancels.
```

As an alternative, you can replace FPTR 2 72 with FPTR 2 74. The differences are that the entry does not save a copy of the original meta on the virtual stack and that instead of the selected object, the index is returned. The indices start at zero, not one.

#### 33.1 The Choose Items meta

meta is a meta object (see Chapter 12) that contains the items which should be shown in the selection box. All object types are allowed, and they will be decompiled for display.

#### 33.2 The Title String

\$title is the title. It will be shown in a small box on top of the choose box. No title will be shown if this is the empty string. This can be useful when the the contents of the choose box do not need a further explanation. Omitting the title makes space for an additional item line.

#### 33.3 The Initially Selected Item

When the choose engine starts, an item is already highlighted. Usually this is the first item, but you can select another one with the #initial parameter. The numbering starts with zero, not one.

#### 33.4 The Message Handler

::message is a program, the message handler. A message handler is a general way to pass a variable number of optional parameters to an application. The application will call the message handler program with different "messages" (normally a bint) in stack level 1, and maybe additional arguments in other stack levels. The handler can decide to handle this message. If it does handle it, it should do its work and return TRUE. If it decides to ignore the message, it should just drop the bint and return FALSE. The empty message handler therefore is the command DROPFALSE (which you can conveniently push in the stack with 'DROPFALSE). When you use the user command xCHOOSE, it just supplies DROPFALSE and hands over to the more general engine.

The message handler can handle the following messages:

Message	Message name and meaning	
BINT1	MsgDispBox	
	This message has to do with the display of the choose box. It	
	currently not well understood. The stack diagram of the messag	
	handler for this message seems to be	
	(#1 $\rightarrow$ ::prog TRUE)	
	( $\#1 \rightarrow FALSE$ )	
BINT2	${f MsgDispTitle}$	
	This should display the title. If not handled, the title is drawn	
	using the supplied \$title argument.	
	$(#2 \rightarrow TRUE)$	
	$(#2 \rightarrow FALSE)$	
BINT3	MsgEndInit	
	This message is executed after the initialization of the choose	
	box, but before control is handed over to the POL.	
	$( #3 \rightarrow TRUE )$	
	$( #3 \rightarrow FALSE )$	
BINT4	MsgKeyPress	
	This is a key handler, similar to the ones used by a POL. When	
	the user presses a key, the message handler is called with the	
	keycode and plane (see section 30.1), and the message BINT4 on	
	the stack. It should return the key definition (an address or a	
	secondary), TRUE and TRUE again.	
	If the key is not handled, FALSE must be returned. Here is the	
	stack diagram for the message handler regarding this message:	
	(#kc #pl #4 $\rightarrow$ KeyDef TRUE TRUE)	
	— yes, TRUE twice!	
	$(\#kc \#pl \#4 \rightarrow FALSE)$	
BINT5	MsgMenu	
	This must return the menu which is shown to the user during	
	the selection. The return value for this message is evaluated to	
	get the menu. The menu is not automatically updated when you	
	move the selection, but message #6 can be used to enforce an up-	
	date. If the menu has more than one page, you must handle the	
	NXT and PREV keys in the keyhandler — they are not handled	
	by default.	
	$(#5 \rightarrow \{ menu list \} TRUE)$	
	(#5 → ::prog_returning_list TRUE)	
	( $\#5 \rightarrow FALSE$ )	

# MessageMessage name and meaningBINT6MsgEndEndDispThis message is called after the redisplay of the choose box finishes (because you changed the selected item). You can use this to force an update of the menu display by setting 24LAM to FALSE. See the example below.(#6 $\rightarrow$ TRUE)(#6 $\rightarrow$ FALSE)

#### 33.5 The Browser and Lams

The browser POL uses 24 unnamed local variables, so maybe you should not rely on unnamed locals yourself. Better use named locals for this purpose. A few important unnamed LAMs used by the browser engine are:

LAM	Contents	
1LAM	Quit. Set this to TRUE if you want the POL to exit.	
2LAM	DispOffset. Index of selected item with respect to DispTop.	
3LAM	DispTop. Index of the first choose item currently visible on the	
	screen.	
17LAM	The message handler.	
14LAM	The redisplay program.	
24LAM	DisplayMenu. Set this to FALSE in order to enforce a redisplay of	
	the menu.	

#### 33.6 Accessing the Selected Item

To use the browser for more than just selecting an item, you must write programs which will be accessible with the key handler or with the menu. On of the most important tasks in these programs is to find out what the current item is. The choose box engine keeps two copies of the choose list on the Virtual Stack, and you can use these to get the current item. On level one of the Virtual Stack, the list is inverted, and the items which have already been shown in the CHOOSE box are converted to strings (sensitive to flag -85). On level three of the Virtual Stack there is a copy of the original list. The index of the current item is available with this code snippet:

```
:: 2GETLAM 3GETLAM #+ ;
```

Indexes start at 0.

To access the current item use one of these methods:

1. Get the decompiled string. This is very fast and only 8 bytes.

```
:: 2GETLAM 3GETLAM #+ GetElemBotVStack ;
```

2. Get the original item. There is no supported way to get to the third level of the Virtual Stack directly, so you have to dig it out and restore the stack afterwards. Here is a way to do it (35 bytes):

```
1 ::
    GetVStackProtectWord PopMetaVStack
    GetVStackProtectWord PopMetaVStack
    2GETLAM 3GETLAM #+
5 GelElemTopVStack
    1LAMBIND
    PushMetaVStack&Drop SetVStackProtectWord
    PushMetaVStack&Drop SetVStackProtectWord
    1GETABND
10 ;
```

This looks complicated, but it is also quite fast and actually used in the ROM for the Help key of the catalog.

3. If you find 2 too long, you can keep a copy of your original list, for example in a named LAM "mylist". If you did that before calling the browser flashpointer, you get the current item with

```
:: LAM mylist 2GETLAM 3GETLAM #+ #1+ NTHCOMPDROP;
```

## 33.7 Saving and Restoring the Screen

If you want to use a menu key or another key to do an excursion from the choose box which uses the display, you must save and restore the current screen around it. This is because the browser POL only updates as little as possible on the display, so when you return from your excursion, the display will look bad and not recover. There are two simple flashpointers which can be used to save and restore the display:

```
FPTR 2 88 Save the current isplay
FPTR 2 89 Restore the saved display
```

Note that these commands use a specific storage place, so they cannot be used by stacked choose boxes (a choose box creating another choose box which needs to save its screen for an excursion). In such cases, you need to save and restore copies of HARDBUFF and HARDBUFF2.

## 33.8 An Example

Below follows an example for the application of the browser engine. This program displays the numbers 1-100 for multiple selection and returns a list of all selected values. Pressing the Squareroot key displays the square root of the current number in a message box. In the menu, pressing F1 adds the decompiled version of the currently selected number (a string!) to the return list. Pressing F2 will show some help text about the choose box. There is another menu button F3 which does not do anything, but which shows if the selected number is even or odd. Since this display changes, we need message six to force a menu update. F5 and F6 are the usual CANCL and OK actions.

Figure 33.1 shows this program while running.

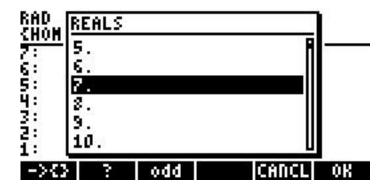


Figure 33.1: The '49 browser example

```
(Empty list to collect)
      NULL{}
      { LAM mylist LAM res } BIND
                                     (Save a copy of the list)
      INNERCOMP
                                     (Explode for FPTR 2 72)
10
      "REALS"
                                     (Title)
                                     (Initial position)
      ::
                                     (The key handler)
        4 OVER#=case
15
          DROP DUP#1= 3PICK 23 #=
          ANDcase
                                     (SQRT key pressed)
          ::
            2DROP
                                     (DROP the keycodes)
20
            ::
              LAM mylist
              2GETLAM 3GETLAM
              #+ #1+
25
              NTHCOMPDROP
                                     (Get current value)
              %SQRT DO>STR
                                     (Compute SQRT)
              FlashWarning
                                     (Display)
            ;
            TrueTrue
                                     (Yes, we handle this key)
30
          FALSE
                                     (Other keys not handled)
        ;
        5 OVER#=case
                                     (Provide a menu)
        ::
35
          DROP
           ' ::
            NoExitAction
                                     (Do not save as LastMenu)
             ("Add to list" menu key)
                 :: TakeOver
40
                   LAM res
                                     (Get current list)
                    2GETLAM
                                     (Get element as string)
                    3GETLAM #+
                    GetElemBotVStack
                                     (Add to list)
                    >TCOMP
45
                    ' LAM res STO
                                     (STO current list)
                 ; }
               { "?"
                                     ("Help" menu entry)
                 :: TakeOver
                    FPTR 2 88
                                     (Save current screen)
50
                    DOCLLCD
                                     (Clear screen)
                    ZEROZERO
                                     (Next is the help text)
```

```
"->{}
            ADD
            HELP
    SORT
            DISP ROOT"
55
                    $>GROBCR
                    XYGROBDISP
                                      (Display help text)
                                      (Wait for any key)
                    WaitForKey
                    2DROP
                    FPTR 2 89
                                      (Restore the screen)
60
                 ;
                 ::
                                      (Button to show)
                                        ("even" or "odd")
                    TakeOver
                    LAM mylist
                                      (The list)
65
                    2GETLAM
                    3GETLAM #+ #1+
                                      (Get current element)
                    NTHCOMPDROP
                    DUP
                                      (Test if even)
                    %2 %/
70
                    %FLOOR
                    %2 %* %= ITE
                    "even" "odd"
                                      (Return correct label)
                 ;
                 NOP
                                      (No action when pressed)
75
               NullMenuKey
                                      (4th key is empty)
                                      (Default CANCL action)
               { "CANCL"
                 FPTR 2 77 }
                                      (Default OK action)
               { "OK"
80
                 FPTR 2 76 }
             }
          TRUE
                                      (Yes, we provide a menu)
85
        6 OVER#=case
                                      (Enforce menu update)
         :: DROP FalseFalse
            24 PUTLAM;
        DROPFALSE
                                      (Other messages)
                                        (are not handled)
90
      FPTR 2 72
                                      (Run the CHOOSE engine)
      ITE
        ::
                                      (DROP current value)
          DROP
                                      (Return list)
          LAM res
95
          TRUE
                                      (Push TRUE)
         ;
```

```
FALSE (CANCL: return FALSE)
ABND (Free local variables)
```

# 33.9 Reference

Addr.	Name	Description
072002	(^Choose3)	( meta $\pm$ title $\pm$ pos ::handler $\rightarrow$ ob T )
		( meta $\pm$ title $\pm$ pos $\pm$ handler $\rightarrow$ F
		The main choose engine.
074002	(^Choose3Index)	( meta $\$$ title $\#$ pos $::$ handler $\rightarrow$
		#idx T )
		( meta $\pm$ title $\pm$ pos $\pm$ handler $\rightarrow$ F
		Same as ^Choose3, but returns the index
		of the selected item instead of the item
		itself. #idx starts at zero.
070002	(^Choose2)	( meta $title pos \rightarrow ob T$ )
		( meta $title \#pos \rightarrow F$ )
		Call Choose3Index with empty message
		handler. This is just
		:: 'DROPFALSE FPTR2
0.77000	(0.5)	^Choose3Index ;
073002	(^Choose3Save)	( meta \$title #pos ::handler $\rightarrow$ ob T )
		( meta $\pm$ title $\pm$ pos $\pm$ handler $\rightarrow$ F
		Save and restore HARDBUFF/2 around a
		Choose3 call.
005002	(^sysCHOOSE)	( $title \{ \} \ sel \rightarrow ob \ l \ )$
		( $title \{ \} \ sel \rightarrow 0 \ )$
		Equivalent to User RPL CHOOSE command.
075002	(^ChooseDefHandler)	( $ ightarrow$ ::handler )
		Pushed the default message handler (the one used by the CAT key) on the stack.

Addr.	Name	Description
088002	(^SaveHARDBUFF)	$( \rightarrow )$
		Save HARDBUFF and HARDBUFF2 is a safe
		place.
089002	(^RestoreHARDBUFF)	$( \rightarrow )$
		Restore HARDBUFF and HARDBUFF2 saved
		with SaveHARDBUFF.
077002	(^Choose3OK)	$( \rightarrow )$
		The OK action executed by Choose3 if OK
		or ENTER is pressed.
076002	(^Choose3CANCL)	$( \rightarrow )$
		The CANCEL action executed by Choose3
		if CANCL or ON is pressed.

# Chapter 34 Using the HP48 Browser

The HP48 browser (which is still present in the HP49) allows you to do many things. Basically, it displays a list of entries, from which you can select one or many (unlike the new HP49 browser, which only allows one item to be selected), and you can act on those entries by means of menu keys or hard key assignments.

This "old" engine has a few features that the HP49 one does not have, such as a full-screen mode. It is, however, more complicated to use. Just like the Input Form engine (see Chapter 35), it has thousands of features, and generally there are several ways to accomplish the same thing.

The browser is called by the entry ~Choose. It expects five parameters in the stack. It will the return the results and TRUE, or just FALSE, depending on the way it was exited (more on that later). Here are the stack diagrams:

```
(::Appl $Title ::Converter {}Items Init \rightarrow result TRUE) or (::Appl $Title ::Converter {}Items Init \rightarrow FALSE)
```

Here, result is either a list or a single object, depending on whether check marks and multiple selections are enabled.

# 34.1 The :: Appl Parameter

This is a program that allows configuration of several aspects of the browser. It works as other message handlers do: it is called with a bint in the stack, representing the code of the message. If the message is handled, the program should return any data required by the message and TRUE, otherwise it returns FALSE. Which means that DROPFALSE (which can be pushed in the stack with the command 'DROPFALSE) is a valid value for this parameter, meaning that no messages are handled, and that default values should be used at all the times.

Here are the descriptions of some of the messages:

Code (Decimal)	Description and Stack
57	Number of lines the browser will display on the screen. The default depends on the current font, and on system flag 90.
	( → # )
58	Height of browser line. Probably this does not need to be changed.
59	( $\rightarrow$ # Width of browser line. Leave space for the display of arrows if the number of elements may be grater than the page size. ( $\rightarrow$ # )
60	Should return TRUE if the browser will be full-screen, or FALSE if windowed. The default is windowed.  ( — flag )
61	Should return TRUE if check marks are allowed, thus supporting the selection of multiple items, or FALSE if not. The default is not to allow check marks.  (
62	Returns the number of elements. If your program changes the number of elements during execution, you must handle this message. $(\rightarrow \#)$
63	Should return the coordinates of the upper left corner of the browser selection box. You probably do not need to change the default value.  ( $\rightarrow \#x \#y$ )
64	This message should return the initial difference be- tween the marked selection and the top of page. Be sure that the difference is less than the current selec- tion and less than the page size, otherwise the calculator
65	may crash. ( $\rightarrow$ # ) This message is called when the background needs to be painted. Its action can be used to draw something else on the background. ( $\rightarrow$ )

Code (Decimal)	Description and Stack
66	This message is called when the title needs to be painted. Its action should draw the title in HARDBUFF. Most of the times, this is not handled, and the title is drawn from the
	\$Title parameter. $(\rightarrow)$
67	Returns title as a grob. Most of the times, this is not handled, and the title is drawn from the $Title$ parameter. ( $\rightarrow$ grob)
68	If message 67 is not defined, this is called to return the title as a grob, but only for full-screen mode.  ( → grob)
69	If message 67 is not defined, this is called to return the title as a grob, but only for windowed mode.  ( — grob )
70	If the \$Title parameter is not a null string, this entry is called to return a title string. This overrides the \$Title parameter. $(\rightarrow \$)$
74	This message should draw all visible lines of the browser. ( $\rightarrow$ )
79	This message should display one line of the browser. If this is the selected line, this message should draw this line in inverse video or mark that it's the selected one in another way. ( $\# \to$ )
80	This message is an alternative to supplying the items as the {}Items parameter. It supplies the number of the item, and this message should returns the item. Any object can be returned; ::Converter will be called to convert this into a string. If you want to have dynamically-changing items in the browser, this message allows that. But message 82 is probably better in this case. ( $\# \to ob$ )
81	This message converts one element into a grob. (This overrides the ::Converter parameter.) If should return a grob with dimensions 7NULLLAMx8NULLLAM. If check marks are enabled, you must incorporate the check mark in the grob if the item is checked. (# \rightarrow grob)

Code (Decimal)	Description and Stack
82	This message is like message 80, but the object is already returned as a string. :: Converter is <i>not</i> called afterwards. If this message is used, you do not need to write a :: Converter.  ( $\# \rightarrow \$$ )
83	Returns a list describing the menu. The format of the list is the same of InputLine and Input Forms, see section 37.1. ( $\rightarrow$ { } )
85	This message is called when the browser is started, after everything has been set. $(\rightarrow)$
86	This is called when an item is checked or unchecked. The default action handles checking and unchecking of items pretty fine, so you probably do not need to handle this message. ( $\# \to$ )
87	This message is called before the browser exits. ( $\rightarrow$ )
91	This is called after the ON key is pressed, or the CANCL menu key. If TRUE is returned, the browser exits. If FALSE is returned, the browser continues.  ( — flag)
96	This is called after the ENTER key is pressed, or the OK menu key. If TRUE is returned, the browser exits. If FALSE is returned, the browser continues. ( $\rightarrow$ flag )

# 34.2 The \$Title Parameter

This parameter specifies the title. There are messages that can override this parameter: 66, 67, 68, 69 and 70.

#### 34.3 The :: Converter Parameter

This is a secondary that converts whichever kind of object is used as a list into a string for display. The stack diagram for this secondary is (ob  $\rightarrow$  \$)

If you handle messages 81 or 82, you do not need to write this program to do the conversion. However, the browser allows the user to press Alpha followed by a letter to search for an object that starts with that letter and jump to it. This requires the ::Converter parameter, even if those messages are provided. So you should ensure this parameter someway returns a string. The DO>STR entry can be of great use here.

## 34.4 The {}Items Parameter

You can specify a list of objects here, or you can specify an empty list, and use messages 80, 81 or 82 to provide the elements.

#### 34.5 The Init Parameter

This can be either a binary integer or a list. If it is the bint 0, the browser works as a viewer, disallowing selections. If it is any other bint, it is the initially selected element.

If multiple selections are enabled, you can specify instead a list of bints, representing the initially checked elements.

## 34.6 Typical Browser Usage

By reading the description of the messages and the parameters above, you have probably noted that there are several ways to provide the element that will form the browseable list, and you may have been confused by that. Here, two ways to do that will be listed.

• You can provide the elements using the {} Items parameter, and provide a :: Converter that will convert one of those elements into a string. You do not need to worry about messages 80, 81 or 82. This method is good if the list of elements will not change while the program is running.

• You can leave the {} Items list empty, and store the list of elements somewhere else (most likely in a lam). Then, use messages 80, 81 or 82 to return the elements. If you use messages 81 or 82, you will return elements already as a grob or as a string, and ::Converter can be a null secondary. Or you can use 80 to return some object, and then use ::Converter to make a string out of it. This method is good if the elements change while the program is running. If you use this technique, you must also handle message 62.

When the number of elements in the browser changes, run this code to adapt the browser to the changes:

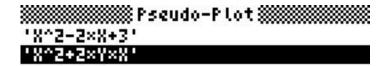
```
1
      ROMPTR 0B3 03E
                             (Re-read # of elements)
      ROMPTR 0B3 026
                             (Re-read width)
      18GETLAM
                             (#Index)
5
      12GETLAM
                             (#NumOfElements)
      DUP#0=IT
        DROPONE
                             (Reduce #index if #NumOfElements)
      #MIN
                             (was reduced)
10
      18PUTLAM
      FALSE ROMPTR 0B3 019 (Recalculate offset)
```

## 34.7 An Example

This example uses the browser to allow the user to enter a list of equations (inspired by the Y= window, but considerably different). Initially, the list is empty. The user then adds equations to the list. Equations can also be edited or deleted.

This program handles messages 62 and 82 to return the number of elements and an equation already converted to a string when asked for it. The equations are stored in a named LAM. Some other messages are also handled to configure other aspects of the browser.

Figure 34.1 shows this program while running.



## Add Del Edit CANCL OK

Figure 34.1: The '48 browser example

```
1 DOBIND
5
      ' ::
                                   (the ::Appl parameter)
        60 #=casedrop TrueTrue
                                   (use full screen)
        62 #=casedrop ::
                                   (number of elements)
          LAM EQS LENCOMP
10
          DUP#0=IT
            #1+
          TRUE
        82 #=casedrop ::
                                   (return nth element as str)
15
          LAM EQS SWAP
          NTHELCOMP
          ITE
            ::
                                   (convert to string)
              setStdWid
20
              FPTR2 ^FSTR7
           "No equations"
          TRUE
25
        83 #=casedrop ::
                           (the menu)
            { "Add"
              ::
                PushVStack&Clear (save stack)
30
                DoNewEqw
```

```
DEPTH
                #0<> IT
                                  (add eqaution)
                  ::
                    LAM EQS SWAP
35
                    >TCOMP
                    ' LAM EQS
                    STO
                    ROMPTR B3 3E (re-read # elements)
40
                PopMetaVStackDROP
              "Del"
              ::
45
                LAM EQS
                INNERDUP
                                  (quit if empty)
                #0=case DROP
                PushVStack&Keep (save stack contents)
                reversym
50
                DROP
                18GETLAM
                ROLL
                DROP
                18GETLAM #1-
55
                UNROLL
                DEPTH
                { }N
                 ' LAM EQS STO
                PopMetaVStackDROP (restore stack)
60
                ROMPTR B3 3E
                                   (re-read # elements)
                18GETLAM
                                   (change selected element)
                12GETLAM
                                   (if necessary)
                #MIN
                18PUTLAM
65
                FALSE ROMPTR B3 19
              "Edit"
              ::
70
                                  (get element)
                LAM EQS
```

```
18GETLAM
                NTHELCOMP
                NOT?SEMI
                                   (quit if empty)
                FPTR2 ^EQW3Edit
                                   (edit)
75
                NOT?SEMI
                                   (quit if not changed)
                 18GETLAM
                LAM EQS
                                   (replace)
                PUTLIST
                 ' LAM EQS STO
80
            }
            NullMenuKey
            { "CANCL" FPTR2 ^DoCKeyCancel }
            { "OK" FPTR2 ^DoCKeyOK }
85
          TRUE
        DROPFALSE
90
      "Pseudo-Plot"
                                   (title)
      ' NULL::
                                   (converter)
      NULL{}
                                   (no items - msgs are used)
      BINT1
                                   (initially selected elt)
95
                                   (run browser)
      ROMPTR2 ~Choose
      ABND
    ;
```

Addr.	Name	Description
0000B3	~Choose	( ::Appl \$Title ::Convert {}
		offset $\rightarrow$ {}' T)
		<pre>( ::Appl \$Title ::Convert {}</pre>
		offset $\rightarrow$ ob T )
		<pre>( ::Appl \$Title ::Convert {}</pre>
		offset $ ightarrow$ 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.
0050B3	~ChooseMenu0	$( \rightarrow \{\} )$
		Menus with "OK".
0060B3	~ChooseMenu1	$( \rightarrow \{\} )$
		Menus with "CANCL", "OK".
0070B3	~ChooseMenu2	$( \rightarrow \{\} )$
		Menus with "CHK", "CANCL", "OK".
0630B3	~ChooseSimple	( $title \{items\} \rightarrow ob T$ )
		( $title \{items\} \rightarrow F$ )
		Simple interface to the HP48 choose
		engine. On the HP49G, calls
		^RunChooseSimple.
004002	^RunChooseSimple	( $title \{items\} \rightarrow ob T$ )
		( $title \{items\} \rightarrow F$ )
		Simple interface to the HP48 choose en-
		gine.
09F002	^DoCKeyCheck	$( \rightarrow )$
		Toggle check on current item.
0A0002	^DoCKeyChAll	$(\rightarrow)$
0.0000	A	Check all elements.
0B0002	^DoCKeyUnChAll	$(\  ightarrow\ )$ Uncheck all items.
09E002	^DoCKeyCancel	Oncheck an items. ( $\rightarrow$ )
096002	Dockeycancer	Simulate Cancel.
09D002	^DoCKeyOK	( → )
	1	Simulate OK.

Addr.	Name	Description
0B3002	^LEDispPrompt	$( \rightarrow )$
		Redraw title.
0B2002	^LEDispList	$( \rightarrow )$
		Redraw browser lines.
0B1002	^LEDispItem	$( \# \rightarrow )$
		Redraw one line.
0150B3	(~BBMoveTo)	$( \# \rightarrow )$
		Moves selection to line and updates dis-
		play.
0190B3	(~BBRecalOff&Disp)	(flag → )
		Recalculates offset of selected item in page,
		and redraws lines if the flag is TRUE.
0220B3	(~BBRunEntryProc)	$( \rightarrow )$
		Sends message 85 to ::Appl, thus running
	,	the user-defined start-up procedure.
0230B3	(~BBReReadPageSize)	$(\rightarrow)$
		Re-reads the size of the page (message 57).
0240B3	(~BBReReadHeight)	$(\rightarrow)$
		Re-reads the height of the browser line
		(message 58).
0250B3	(~BBReReadCoords)	$(\rightarrow)$
		Re-reads the coordinates of the browser
		box (message 63).
0260B3	(~BBReReadWidth)	$( \rightarrow )$
		Re-reads the width of the browser line
		(message 59).
0280B3	(~BBRunENTERAction)	$(\rightarrow)$
		Sends message 96 to ::Appl, thus running
		the OK action. It does not check the value
		returned and never exits.
0290B3	(~BBRunCanclAction)	
		Sends message 91 to ::Appl, thus running
		the CANCEL action. It does not check the
00=0=0	(	value returned and never exits.
02F0B3	(~BBReDrawBackgr)	$(\  ightarrow\ )$
027052	(~DDG_+NG1-)	Redraws the background.
0370B3	(~BBGetNGrob)	$( #n \rightarrow grob )$
		Returns nth element as a grob.

Addr.	Name	Description
0380B3	(~BBGetNStr)	( $\#n \rightarrow \$$ )
		Returns nth element as a string.
03B0B3	$(^{\sim}$ BBRereadChkEnbl $)$	$( \rightarrow )$
		Re-reads whether checkmarks are en-
		abled. (Message 61).
03C0B3	(~BBRereadFullScr)	$( \rightarrow )$
		Re-reads whether to use full-screen mode.
		(Message 60).
03D0B3	(~BReReadMenus)	$( \rightarrow )$
		Re-reads the menu. (Message 83).
03E0B3	(~BBReReadNElems)	$( \rightarrow )$
		Re-reads the number of elements. (Mes-
		sage 62).
03F0B3	(~BBGetN)	$( #n \rightarrow ob )$
		Returns nth element.
04B0B3	(~BBIsChecked?)	$( \#n \rightarrow flag )$
		Returns whether the given element is
		checked.
0520B3	(~BBUpArrow)	$(\rightarrow grob)$
		Returns up arrow as grob
0530B3	(~BBDownArrow)	$(\rightarrow \text{grob})$
054053	(~~~~)	Returns down arrow as grob
0540B3	(~BBSpace)	$( \rightarrow \text{grob} )$
050003	(~DDD~Dovm)	Returns a space as grob. $(\rightarrow)$
0590B3	(~BBPgDown)	Go down one page.
05A0B3	(~BBPgUp)	( $\rightarrow$ )
OJAODJ	( DDF GOP)	Go up one page.
05B0B3	(~BBEmpty?)	( → flag )
002020	( 222	Returns TRUE if the browser has no ele-
		ments.
05C0B3	(~BBGetDefltHeight)	$( \rightarrow \# )$
	,	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.
0190E0	~BRRclC1	$( \rightarrow )$
		:: LAM 'BR5 ;

# 34.8.1 NULLLAMs Used by the Browser

The browser uses a great number of unnamed lams to store its information. Here is a description of them:

Lam	Description	Type
1	Used by CACHE	n/a
2	POL exit condition	flag
3	Initial display status. This is a list in this format:	{}
	{ DA1IsStatFlag DA2bEditFlag DA1BadFlag	
	DA2aBadFlag DA2bBadFlag DA3BadFlag }	
4	Menu before browser was run	grob 131x8
5	Screen before browser was run	grob 131x56
6	Offset in page	#
7	Height of browser line	#
9	x coordinate of upper left corner of browser in	#
	HARDBUFF	
10	y coordinate of upper left corner of browser in	#
	HARDBUFF	
11	Page size	#
12	Number of elements	#
13	Menu	{ }
14	Full screen?	flag
15	List of indexes of checked items	flag
16	Check marks enabled?	flag
17	TRUE if is a browser, FALSE if it is a viewer	flag
18	Current selected index	#
19	{}Items	{}
20	::Converter	::
21	\$Title	\$
22	::Appl	::

# Chapter 35 Creating Input Forms

Input forms provide a graphical interface for entering data required by a program. Data is entered by means of several fieds, which are "spaces" that the user can fill with the apropriate data. Input forms are used in many places in the HP49. You can see one by pressing the MODE key.

It is possible to create input forms in User RPL, with the INFORM command, but this is not one of the easiest tasks. In System RPL, it is even more difficult. But there are several advantages: in User RPL, you can only have text fiels, in System RPL you can have check boxes or choose fields. You can also restrict the valid inputs, and make fields appear or disappear during the execution. Finally, in System RPL the input forms are considerably faster.

Input forms are created with the *lemain* command, which is a flash-pointer. It needs lots of arguments. They are divided in three categories: label definitions, field definitions, and general information. Each label and field definition is composed of several arguments.

The <code>lfMain</code> command referes to the new input form engine present in the HP49. The old HP48 engine is still present (and has had some speed improvements); the old <code>DoInputForm</code> command is still present and the forms created based on that command will still work. The arguments are the same for both entries, but the message handling (see section 35.4 below) has changed. There are a few incompatibilities between both engines.

The table below shows the general argument structure for the ^IfMain command:

Parameter	Description
label_1	
	Label definitions
label_n	
field_1	
• • •	Field definitions
field_n	
#labels	Number of labels

Parameter	Description
#fields	Number of fields
MessageHander	See section 35.4 below
Title	Title to be shown on top of screen

### 35.1 Label Definitions

Each label definition consists of three arguments:

Parameter	Description
label_text	Text to be displayed
#x_offset	X coordinate
#y_offset	Y coordinate

label\_text is a string, that will be converted to a grob using the minifont. This text will be displayed at the specified coordinates. These are two bints representing the x and y positions of the label in the screen. The top-left corner has coordinates (0, 0), and coordinates increase down- and right-wards.

The new input form engine also supports a grob as argument, this grob will be directly displayed at the given coordinates.

## 35.2 Field Definitions

Each field definition consists of thirteen arguments:

Parameter	Description
MessageHandler	See section 35.4 below
#x_offset	X coordinate
#y_offset	Y coordinate (normally label Y coordinate - 1)
#Length	Length of field
#Height	Height of field (usually 8)
#FieldType	Type of field, see below for valid values.
#AllowedTypes	List of valid object types
Decompile	See below
"HelpString"	Help string
ChooseData	See below
ChooseDecompile	See below

Parameter	Description	
ResetValue	Reset value	
InitValue	Initial value	

The message handler will be described below.

The x and y positions specify where the field will appear. They work similarly to the x and y positions of label definitions. Then length and height are also two bints, which specify the size of the field.

The field type is a bint which defines the type of the field:

Decimal value	Field Type
1	Text field: user can enter anything.
23	Extended text field (DoInputForm engine only): The user
	can enter anything, or select a variable using the filer.
12	Choose field: user must select from a list of valid values.
2	Combo field: user can select from a list of values or enter
	another.
32	Checkbox field.

The allowed types parameter is used in the text, extended text and combo fields: it is a list of bints, representing the allowed types of objects that can be entered in that field. You can find the object types in the table of section 29.2. You have to use the values in the "User Type" column, as bints. Other fields should specify MINUSONE. You can also specify MINUSONE for text and combo fields, this means that all kinds of objects are accepted. In the extended text field, the list of types is also used to limit the variables displayed in the filer.

Decompile is a bint that specifies how the entered objects should be displayed in the screen. Its meaning depends on the bits that are set. First, you should start with BINT2 or BINT4: the former tells that numbers will be decompiled using the current mode, the latter specifies that STD mode should be used. If the field will not hold numbers, it does not make much difference in which value you choose.

After you have specified the basic way to decompile objects, you can also set some flags to configure it further. If you want to use the minifont when displaying the field valued, add 1 to the value. If you add 8 to the value, then only the first character of the string will be displayed. If the object the field holds is a list (or another composite, as a matter of fact), you can add 16 or 32, to get the first or second object of this composite, respectively, and display

this object according to the rules defined by the other values. This option is sometimes useful when using choose fields, but not for normal text fields.

Note: DoInputForm does not support fiels decompiled with the minifont.

You can also specify the Decompile paramater as BINTO. If this is done, no decompilation is done: you can only use strings in the field, and they will be displayed, without the quotes, in the normal font.

The next parameter specifies the help string that will be shown in the last line of the display when that field has the focus. Enter anything you want.

The ChooseData parameter is only used in list and combo fields. Other types should have MINUSONE as this parameter. This paramer is the list of values that will be presented to the user for selecting. When you use a decompile value that includes the value 16, you can use a list like this:

```
{ { "label1" <foo> } { "label2" <bar> } { ... } ... }
```

This way, only the first objects will be shown, but the entire list will be returned. (Like the INFORM User command does.)

When using DoInputForm (but not ^IfMain, you can also specify a string in the ChooseData parameter of text fields. This means that the text field will allow the user to browse the variables stored in memory and use the contents of some variable as the value of the field.

Apparently, ^IfMain ignores the ChooseDecompile parameter. Just specify it with the same value you used for the Decompile parameter.

The reset and initial values are the contents of the field that are shown when the form is initially displayed, and when it is reset. It should be an object of the types allowed for that field, for list fields it will be one of the elements of ChooseData list. For check fields, use TRUE or FALSE. You can leave text or combo fields empty by specifying MINUSONE as one or both of this parameters.

## 35.3 Label and Field Counts

These are two bints, representing the number of labels and fields defined. Note that since they are different values, you can have labels which just show some kind of information to the user, or fields without any label definition.

## 35.4 Message Handlers

As with other input/output applications of the HP49, input forms use message handlers to allow the programmer to have more control over the input form. There is one message handler for each field, and one for the input form itself. The messages are passed whenever something "interessant" happens to a field or the input form, and during the initialization of the input form.

As with other message handlers, the program you provide is called with a message number (a bint) in level one, and sometimes other parameters. If the program handles the message, then it should return whatever is required by the specific (sometimes nothing). If the message is not handled, it should drop the message number and push FALSE in the stack, leaving any other arguments there. So, a message handler that handles no messages is simply DROPFALSE, which, as you know, can be conveniently pushed in the stack with 'DROPFALSE.

In the message handling, the entries listed in the reference section below can be used to retrive information from the input form or to modify it.

Section 35.8.2 will describe each of the available messages in  $^{^{1}IMain}$ . The messages of DoInputForm are different.

Here is a template message handler program if only one message is handled:

```
1 '::
    IfMsgGetFocus (or any other message)
    #=case
    ::
5 * Here is the message handling code
        TRUE (to tell the system the message was handled)
    ;
    FALSE (indicate that other messages were not handled)
;
```

And this is a template message handler for two or more messages:

35.5. The Title 253

```
::
    * Code for message.
10 ;
    * And possibly more.

    DROPFALSE (other messages are not handled)
15 ;
```

## 35.5 The Title

This is a string that will be shown on the top of the display, with the small font. If it is longer than 32 characters (the width of the screen), it will be truncated and "..." will be appended.

With <code>^IfMain</code>, instead of a string you can provide your own grob to be displayed. It should have the size of 131x7 pixels.

# 35.6 Results Of The Input Form

The stack output, if the user exited the input form by ENTER is:

```
N+1: field_1
N: field_2
...
2: field_n
1: TRUE
```

If CANCEL was used to exit the form, then just FALSE is returned.

The value of each field depends on the types allowed for that field, and on the way the possible values of list fields are specified. If a field is empty, <code>xNOVAL</code> is returned.

## 35.7 An Example

This example imitates the HP49 transfer dialog, but far from completely. There are many differences, and this example has, naturally, no functionality beyond displaying an Input Form.

The code defines all the labels and fields, and the input form has a simple message handler that handles two messages: one message to set the field that will start with the focus, and one to configure the last three softkeys to look like the ones in the Tranfer dialog. (Our keys, however, only beep when pressed...)

Figure 35.1 below displays the screen when this program is run.



Figure 35.1: The Input Form example

```
1
    ::
    * Label definitons
       "Port:"
                 1
                     10
       "Type:"
                 70
                     10
5
                     19
       "Name:"
                 1
       "Fmt:"
                 1
                     28
       "Xlat:"
                     28
                 49
       "Chk:"
                 104 28
       "Baud:"
                 1
                     37
10
       "Parity:" 49
                     37
       "OvrW"
                 111 37
    * Field definitions
       'DROPFALSE
                                 (Message handler)
15
       26 9 24 8
                                 (Position & size)
      BINT12
                                 (Field type: choose)
      MINUSONE
                                 (Types, does not apply here)
      BINT0
                                 (No decompilation)
       "Choose transfer port"
                                 (Help text)
20
       { "Wire" }
                                 (Possible options)
      BINT0
                                 (ChooseDecompile - ignored)
       "Wire" DUP
                                 (Initial & reset values)
```

```
'DROPFALSE
25
      92 9 36 8
      BINT12
      MINUSONE
      BINT0
      "Choose type of transfer"
30
      { "Kermit" "XModem" }
      BINT0
      "Kermit" DUP
      'DROPFALSE
                                (Message handler)
35
      25 18 103 8
                                (Position & size)
      BINT1
                                (Field type: text field)
      { BINT5 BINT6 }
                                (Allows ids and lists)
      BINT2
                                (Decompile with stack appearance)
      "Enter names of vars to transfer"
                                            (Help text)
40
      MINUSONE
                                (ChooseDate - n/a)
      MINUSONE
                                (ChooseDecompile - ignored)
      MINUSONE DUP
                                (Initially empty)
      'DROPFALSE
45
      20 27 18 8
      BINT12
      MINUSONE
      BINT0
      "Choose transfer format"
50
      { "Bin" "ASC" }
      BINT0
      "Bin" DUP
      'DROPFALSE
55
      74 27 24 8
      BINT12
      MINUSONE
      BINT0
      "Choose character translations"
60
      { "None" "Newl" "\8D159" "\8D255" }
      BINT0
      "\8D255" DUP
      'DROPFALSE
65
      122 27 7 8
      BINT12
      MINUSONE
      BINT0
```

```
"Choose checksum type"
 70
       { "1" "2" "3" }
       BINT0
       "3" DUP
       'DROPFALSE
75
       20 36 24 8
       BINT12
       MINUSONE
       BINT0
       "Choose baud rate"
80
       { "1200" "2400" "4800" "9600" "15300" }
       BINT0
       "9600" DUP
       'DROPFALSE
85
       74 36 24 8
       BINT12
       MINUSONE
       BINT0
       "Choose parity"
90
       { "None" "Odd" "Even" "Mark" "Spc" }
       BINT0
       "None" DUP
       'DROPFALSE
95
       104 36 ZEROZERO
       BINT32
       MINUSONE
       "Overwrite existing variables?"
100
       MINUSONE
       DUP
       TrueTrue
       9 9
                                 (Number of labels & fields)
105
                                 (InputForm message handler)
         BINT7 OVER#=case ::
                                 (Sets initially focused field)
           DROP
           TWO
           TRUE
110
         BINT12 OVER#=case :: (Configures menu softkeys)
           DROP
           { { "RECV" DoBadKey }
```

# 35.8 Reference

# 35.8.1 Inputform

Addr.	Name	Description
020004	^IfMain	( l1ln f1fm #n #m msg \$ $ ightarrow$
		oblobn T )
		( l1ln f1fm #n #m msg \$ $ ightarrow$
		F)
		l = \$ #x #y
		f = msg #x #y #w #h #type legal
		dec \$hlp ChDat ChDec res init
		Starts an input form using the new
		engine.
2C371	DoInputForm	( l1ln f1fm #n #m msg \$ $ ightarrow$
		oblobn T)
		( l1ln f1fm #n #m msg \$ $\rightarrow$
		F )
		1 = \$ #x #y
		f = msg #x #y #w #h #type legal
		dec \$hlp ChDat ChDec res init
		Starts an input form using the old
005050	~	engine.
0050B0	~IFMenuRow1	$( \rightarrow \{\})$
		Returns the menu for the first menu
		row of an InputForm.

Addr.	Name	Description
0060B0	~IFMenuRow2	( → {} )
		Returns the menu for the second menu
		row of an InputForm.
021004	^IfSetFieldVisible	( $\#$ T/F(fld/lbl) T/F(val) $\rightarrow$ )
		( # T/F(fld/blb) #0 $\rightarrow$
		T/F(val) )
		Toggles the field or label visible or
		invisible. Second argument specifies
		if # means a field or a label. Third
		argument is the value to set. ZERO as
		third argument means to retrieve the
022004	^	current setting.
022004	^IfSetSelected	( $\#$ T/F(fld/lbl) T/F(val) $\rightarrow$ ) ( $\#$ T/F(fld/blb) $\#$ 0 $\rightarrow$
		( # 1/F(11d/D1D) #0 → T/F(val) )
		Toggles the field or label selected or
		not selected (appears in inverse video
		on the screen).
023004	^IfSetGrob	( $\# T/F(fld/lbl) grb \rightarrow$ )
		Sets the grob of a field or a label (mod-
		ifies the data saved in the data string).
024004	^IfSetFieldValue	( val $\#$ $\rightarrow$ )
		Sets the value of a field (full handling,
		including GROB setting).
026004	^IfGetFieldValue	$( \# \rightarrow val )$
		Gets the value of the Nth field.
027004	^IfGetCurrentFieldValue	$(\rightarrow)$
005004		Gets the value of the current field.
025004	^IfSetCurrentFieldValue	$(val \rightarrow )$ Sets the value of the current field.
028004	^IfGetFieldMessageHandler	( $\# \to \text{prg}$ )
020004	ligetrieldmessagehandler	Retrieves a field message handler.
029004	^IfGetFieldType	( $\# \rightarrow \#$ type )
	7.	Retrieves the field type.
02A004	^IfGetFieldObjectsType	( # → {} )
		Retrieves the field object type list.
02B004	^IfGetFieldDecompObject	( $\#$ $\rightarrow$ val )
		Retrieves the field decomp value.

Addr.	Name	Description
02C004	^IfGetFieldChooseData	( # → {} )
		Retrieves the field data for choose.
02D004	^IfGetFieldChooseDecomp	( $\#$ $\to$ val )
		Retrieves the field decomp value in
		case of choose.
02E004	^IfGetFieldResetValue	$( \# \rightarrow val )$
		Retrieves the field reset value.
02F004	^IfSetFieldResetValue	( val # → )
		Changes the field reset value.
030004	^IfGetFieldInternalValue	$( \# \rightarrow \text{val} )$
		Retrieves the field internal value.
031004	^IfDisplayFromData	$(\rightarrow)$
		Displays the datastring on the screen.
		Takes care of the command line size.
032004	^IfGetNbFields	$(\rightarrow \sharp n)$
		Recalls the number of fields from the
000004		data string.
033004	^IfCheckSetValue	$($ # val $\rightarrow$ $)$
004004		Checks or uncheck a check field.
034004	^IfCheckFieldtype	$( ob \rightarrow ob flag )$
		Checks if an object meets the current
0.4.00.0.4		field type requirements.
04C004	^IfGetPrlgFromTypes	$( \{\} \rightarrow \{\}' )$
		$( \#FFFFF \rightarrow \#0 )$
		Generates a list of the allowed prologs
025004	0.7.57	for a field.
035004	^IfReset	$(\rightarrow)$
		Resets all fields, set as the current
		value their reset value. Used to ex-
		plode the datalist on the stack to work
026004	^**************************************	on it.
036004	^IfSetField	( $\#$ $\to$ ) Makes a different field "current".
027004	^ T E I	
037004	^IfKeyChoose	$(  ightarrow  ext{val} ) \ (  ightarrow )$
		` ,
		If the current field is a choose field, dis-
		plays the posibilities and let the user
		choose. A value is returned only if the
		user does not press CANCEL.

Addr.	Name	Description
038004	^IfKeyEdit	( → (cmd line) ) Edits the current field value if possible. You cannot edit a choose and a label choose field.
039004	^IfKeyTypes	<ul> <li>( → (cmd line) )</li> <li>( → )</li> <li>Displays a Choose box with all the possible types for this field. A command line is opened only if the user replies with OK.</li> </ul>
03A004	^IfKeyCalc	( $\rightarrow$ val ) Puts the value of the field on the stack and HALT. Allows to the user to com- pute a new value.
03B004	^IfKeyInvertCheck	$(\  ightarrow\ )$ Inverts the current check field value.
03C004	^IfONKeyPress	( → ) On Key handler. Gives the oportunity to the user to perform his own pro- gram. Asks to the IF if we can leave. If Yes, puts a FALSE (quit with ON (if canceled)) and sets the 'Quit LAM to TRUE.
03D004	^IfEnterKeyPress	( → ) Enter Key management. Gives the oportunity to the user to perform his own program. Asks to the IF if we can leave. If yes, puts the fields values on the stack put a TRUE (if validated) and sets the 'Quit LAM to TRUE.
03F004	^IfSetHelpString	( \$dat #n \$/# → \$dat' ) Sets the help string associated with a field. This is used by the automatic IF generator program and should not be use in other ways.

Addr.	Name	Description
040004	^IfSetTitle	( \$dat grb/\$/# → \$dat' ) Alters a DataString modifying the Title part. This is used by automatic IF generator program ans should not be use in other ways.
04A004	^IfInitDepth	<ul> <li>( → )</li> <li>Initializes the internal depth counter.</li> <li>This has to be used when running a command modifying the stack</li> </ul>
042004	^IfMain2	( $dat handl \{\} \rightarrow F $ ) ( $dat handl \{\} \rightarrow oblobn T$ )
		Internal Inform Box main program. Alters a DataString modifying the Title part. This is used by automatic IF generator program ans should not be used in a different way.
043004	^IfPutFieldsOnStack	( $\rightarrow$ oblobn ) Puts on the stack the external value of each field.
044004	^IfSetFieldPos	( # T/F(fld/lbl) #x #y #w #h → ) Changes the size and position of an object Note: You can not change the size or the X position of a label or a check field.
045004	^IfGetFieldPos	( $\#$ T/F(fld/lbl) $\rightarrow \#x \#y \#w \#h$ ) Gets the size and position of an object.
047004	^IfSetAllLabelsMessages	( $\$$ dat bmsg $\#$ n $\to$ $\$$ dat ) Sets the text of a set of labels.
048004	^IfSetAllHelpStrings	( $dat bmsg #n \rightarrow dat$ ) Sets the Help String of all fields.
04D004	^IsUncompressDataString	( \$dc → \$dat ) Uncompresses a compressed data string.

#### 35.8.2 Input Form Messages

The names of the messages are DEFINEs for the numbers. You will find this DEFINEs in the inputform.h file in the include subdirectory.

#### 35.8.2.1 IfMsgKeyPress — 0

This message is sent after each keypress, first to the active field, then to the input form. If the field handles the message, the normal input form key handling is *not* executed.

**Input** 2: #KeyPlane

1: #KeyCode

Output (if handled) 2: ::Key\_Handler\_Program

1: TRUE

Output (if not handled) 3: #KeyPlane

2: #KeyCode 1: FALSE

#### 35.8.2.2 IfMsgLooseFocus — 1

This is sent to a field when it is about to loose the focus. You can do anything here, including taking back the focus. If this is done, then no IfMsgGet-Focus message will be sent to this field.

Input 1: #Field\_That\_Will\_Get\_FocusOutput 2: #Field\_That\_Will\_Get\_Focus

1: TRUE or FALSE

#### 35.8.2.3 IfMsgNewField — 2

This message is sent to the IF just before a new field receives the focus. There is no input, and the output can be either TRUE or FALSE.

#### **35.8.2.4 IfMsgGetFocus** — **3**

This message is sent to the field that has just received the focus. There is no input, and the output can be either TRUE or FALSE.

#### 35.8.2.5 IfMsgGetFieldValue — 4

This message is sent to the current field. It has as input the internal data of the field, and this message can be used to return the external value (which is displayed in the screen). Using this and the IfMsgSetFieldValue messages, it is possible, for example, to store only an offset to the current element when you have a list of fixed values, instead of the actual element.

Input 1: Internal value Output (if handled) 2: External value

1: TRUE

Output (if not handled) 2: Internal value

1: FALSE

#### 35.8.2.6 IfMsgSetFieldValue — 5

The complimentary message of IfMsgSetFieldValue: it gives as input the "external" (or user) value, and the internal value should be returned. If you want a message to be called after each change in the value of a field, this is the one. You can leave the value given as input unchanged, naturally.

Input 1: External value Output (if handled) 2: Internal value

1: TRUE

Output (if not handled) 2: External value

1: FALSE

#### 35.8.2.7 IfMsgGetFieldGrob — 6

This message is sent to the current field. If you decide to handle it, you will have to set the grob that is displayed in the field (you can use the <code>^IfSetGrob</code> entry for this). If you do so, then the standard code of the Input Form that would do this is not called.

Input 2: #Field

1: Value

Output 2: #Field

1: TRUE or FALSE

Here is an example of handling this message:

```
1 ::
        OVER TRUE ROT SWAP (Number, number, TRUE, value)
        $>grob
        FPTR2 ^IfSetGrob
5 ;
```

#### 35.8.2.8 IfMsgSetFirstField — 7

This message is sent during initialization to the input form handler, to get the number of the first field that will be selected. It makes no difference wheter you return TRUE or FALSE, just change the number if desired.

Input 1: #Field Output 2: #Field

1: TRUE or FALSE

#### 35.8.2.9 IfMsgFieldReset — 10

This message is sent to a field that is going to be reset. It is possible to modify the value of the field, if desired.

Input 1: Value

Output 2: Value, possibly modified

1: TRUE or FALSE

#### 35.8.2.10 IfMsgGetMenu — 11

This message is sent to the input form handler during initialization, and can be used to provide a menu for the input form. The menu is in the format described in section 37.1.

Input 1: Menu

Output (if handled) 3: Original menu

2: New menu

1: TRUE

Output (if not handled) 2: Original menu

1: FALSE

#### 35.8.2.11 IfMsgGet3KeysMenu — 12

This message can be used to change the last three softkeys of the first row of the standard input form menu. If handled, it should return a list with three sub-lists, each being a key definition.

Input None
Output (if handled) 2: List
1: TRUE
Output (if not handled) 1: FALSE

#### 35.8.2.12 IfMsgCancel — 13

This allows the user to replace the default quit handler. This message is called when the ON key or the CANCL softkey are pressed. If it is handled, then no standard code is run. The user should alter the value of LAM 'Quit to indicate the POL that the input form should be ended.

Input None
Output (if handled) 1: TRUE
Output (if not handled) 1: FALSE

#### 35.8.2.13 IfMsgCancelKey — 14

This message is sent to the input form handler when the user requests the input form to end via the CANCEL key. The programmer can prevent the input form to end if there is invalid input, for example.

Input None

Output (if handled) 2: TRUE or FALSE

1: TRUE

Output (if not handled) 1: FALSE

When the message is handled, a TRUE in level two means that the input form should end, FALSE means it should continue.

#### 35.8.2.14 IfMsgOK — 15

This is similar to the IfMsgCancel message, but for the OK softkey or  ${\sf ENTER}$  key.

#### 35.8.2.15 IfMsgKeyOK — 16

This message is sent to the input form handler when the user requests the input form to end via the OK key. The programmer can prevent the input form to end if there is invalid input, for example.

Input None

Output (if handled) 2: TRUE or FALSE

1: TRUE

Output (if not handled) 1: FALSE

When the message is handled, a TRUE in level two means that the input form should end, FALSE means it should continue.

#### 35.8.2.16 IfMsgChoose — 17

When the user presses the CHOOS softkey in a choose field, this message is sent, first to the field, and then to the input form (if it was not handled by the field). If it is handled by either, then no standard code is run, and you have to display the choose box yourself.

There are no arguments, and you should return TRUE to prevent the standard code to be executed if you desire that, after having displayed your choose box.

#### 35.8.2.17 IfMsgType — 18

This message is sent to the input form when the TYPES softkey is pressed. If it is handled, no standard code is executed.

There are no arguments, and you should return TRUE to prevent the standard code to be executed if you desire that, after having displayed your choose box.

#### 35.8.2.18 IfMsgCalc — 19

This message is sent to the form when the CALC softkey is pressed. If it is handled, no standard code is executed.

There are no arguments, and you should return TRUE to prevent the standard code to be executed if you desire that, after having displayed your choose box.

#### 35.8.2.19 IfMsgNewCommandLine — 20

This message is sent to the input form when a new command line is created. The system does not care if the message is handled or not. It is just to give the programmer the opportunity to perform anything he needs. There are no inputs, and the output is just TRUE or FALSE, without any difference.

#### 35.8.2.20 IfMsgOldCommandLine — 21

This message is sent to the input form when a command line is cancelled. See message IfMsgNewCommandLine above for more details.

#### 35.8.2.21 IfMsgCommandLineValid — 22

This is sent to a field when the command line is validated.

**Input** None; a command line is present

Output (if handled) No command line; elements in the stack and

TRUE

Output (if not handled) 1: FALSE

#### 35.8.2.22 IfMsgDecompEdit — 23

This is sent to a field when an object needs to be decompiled for editing.

Input 1: Object Output (if handled) 2: String

1: TRUE

Output (if not handled) 1: FALSE

#### 35.8.2.23 IfMsgNextChoose — 24

This message is sent to a choose field when the +/- key is pressed. If it is handled, then the default action is not run. There are no inputs and no outputs, except for the TRUE/FALSE.

#### 35.8.2.24 IfMsgEdit — 25

This is sent to a field when the EDIT softkey is pressed. The input is the current value of the field, the output can be nothing, a modified command line, something in the stack, or a modified field. If this message is handled, then the default code is not run.

# **Chapter 36 The Display**

There are two screens available to the programmer while programming in System RPL: the graphics screen, which is visible, for example, in the Plot application (and referred as PICT in User RPL), and the text screen, which is the graphic visible in the standard stack environment. Whenever possible, the latter should be used, leaving the graphics screen untouched, because that is supposedly a user resource, which should not be changed by programs.

## 36.1 Display Organization

The HP49 system RAM contains three dedicated graphic objects (subsequently called grobs) used for display purposes. The commands below return each of this grobs:

Command	Grob
ABUFF	Text grob (stack)
GBUFF	Graphics grob (PICT)
HARDBUFF	Either the text or gaphics grob, whichever is active.
HARDBUFF2	Menu labels

One thing to note is that the words above return just pointer to the grob, so if you alter the grob, the display will also be altered automatically. Most of the times that is the desired behavior, but if you do not want that, call TOTEMPOB after using any of the words above to make a unique copy in temporary memory. See section 24.1.4 for more information on temporary memory and object references.

The text and graphic grobs may be enlarged, and may be scrolled. The menu label grob has a fixed size of 131x8 pixels.

The command TOADISP makes the text grob visible, and the command TOGDISP makes the graphic grob visible.

The text grob is divided in three regions. The display areas are numbered one, two and three. In many words you will find "DA", which means "Display Area". Figure 36.1 shows each of this areas.



Figure 36.1: The Display Areas

Display area 2 is actually divided in two areas: 2a and 2b. Normally, only area 2a is visible, and it occupies the whole DA 2.

## 36.2 Preparing the Display

Two words establish control over the text display: RECLAIMDISP and ClrDA1IsStat. The first does the following:

- Assures the current display is the text one;
- Clears the text display;
- If necessary, resizes the text display to the default size of 131x56 pixels.

This word works very similarly to the user word CLLCD, the difference is that CLLCD never resizes the text display.

The word ClrDAlIsStat is optional, but most of the time it should be used. It suspends the ticking clock display temporarily. Most graphical programs would not want to have that clock displayed.

When the menu is not necessary, use the word TURNMENUOFF to hide the menu and enlarge the text grob to 131x64 pixels. It is turned on again with TURNMENUON. For more details on the menu, see Chapter 37.

The suggested template for an application that uses the text display is:

## 36.3 Controlling Display Refresh

In some programs, it is desired that, after the application ends, the screen is not redrawn, but continues frozen so that the user can see the results, like the User RPL the command FREEZE does. Other times, it is desired that the display is returned back to normal. In System RPL, several words serve those purposes. The most used ones are listed below; the whole list is in the reference section below.

Word	Action
SetDA1Temp	Freezes display area 1.
SetDA2OKTemp	Freezes display area 2.
SetDA3Temp	Freezes display area 3.
SetDA12Temp	Freezes display areas 1 and 2.
SetDAsTemp	Freezes the whole display.
ClrDA10K	Redraws display area 1.
ClrDA2OK	Redraws display area 2.
ClrDA30K	Redraws display area 3.
ClrDAsOK	Redraws the whole display.

## 36.4 Clearing the Display

The following words clear HARDBUFF, entirely or in part. Remember that HARDBUFF refers to the currently displayed grob, either the text or the graph display. Except from BLANKIT, no words take or return arguments.

Word	Action
CLEARVDISP	Clears entire HARDBUFF.
BlankDA1	Clears display area 1.
BlankDA2	Clears display area 2.
BlankDA12	Clears display areas 1 and 2.
Clr16	Clears top 16 rows.
Clr8	Clears top 8 rows.
Clr8-15	Clears rows 8 to 15 (second status line).
CLCD10	Clears status and stack area.
CLEARLCD	Clears entire display.
BLANKIT	( $\#$ start_row $\#$ rows $\rightarrow$ )
	Clears #rows from HARDBUFF.

### 36.5 Displaying Text

There are two fonts in the HP49: the "system font" and the "minifont". Both can be changed by the user, but it is only possible to access two fonts at each time. The height of the system font (or of its characters, to be precise) can vary, but its characters are always five pixels wide. The size of the minifont is fixed: each character is 3x5 pixels.

There are commands to display text in the system font directly, but not for the minifont. In the latter case, it is necessary to convert the text into a grob and display the grob. The list below only describes the most used ones, for a complete list see the reference section below.

#### 36.5.1 System Font

To display text using the system font, use the commands DISPROW1, DISPROW2... to DISPROW10, which take a string as argument and display it in the specified line of the display. Note that, depending on the size of selected system font and wheter the menu is displayed, some of these commands may not be used. You can always safely display text on the first seven lines, even with the largest system font.

#### **36.5.2** Minifont

As said above, displaying text with the minifont is more complicated. First, put a string in the stack and run the command \$>grob. This will return a grob representing with the string in the minifont. You now need to display this grob on the screen. You can use GROB! or XYGROBDISP for that. For more information on these words and for a general treatment of grobs, turn to Chapter 15. In this same chapter, you will find some other commands which might be more convenient for displaying text with the minifont.

#### 36.5.3 Displaying Warnings

The word FlashWarning is used to display a warning message. It beeps, and then displays the given string in a message box. The user must press OK in order to continue.

Instead of FlashWarning, one can use FlashMsg, which displays the text in the status line, and does not beep. To display a message in the status area, it uses the word DISPSTATUS2, which takes a string with a line break in it, and displays it using the two lines of the status area. After a short pause, the display is returned to the state it was before and the program continues.

#### 36.6 Reference

#### 36.6.1 Display Organization

Addr.	Name	Description
26166	TOADISP	$(\  ightarrow\ )$
		Sets the text display as the active.
2616B	TOGDISP	$( \rightarrow )$
		Sets the graphic display as the active.
25FA4	ABUFF	( $ ightarrow$ textgrob )
		Returns the text grob to the stack.
26076	GBUFF	( $ ightarrow$ graphgrob )
		Returns the graphic grob to the stack. The HP49
		extable address for ExitAction! is the same,
		but this must be a bug.
		extable address for ExitAction! is the same,

Addr.	Name	Description
2608F	HARDBUFF	( $ ightarrow$ dispgrob )
		Returns the current grob to the stack.
26094	HARDBUFF2	( $ ightarrow$ menugrob )
		Returns the menu grob to the stack.
25EDE	HARDHEIGHT	( $ ightarrow$ #height )
		Returns the height of HARDBUFF.
25ED5	GBUFFGROBDIM	( $ ightarrow$ #height #width )
		Returns dimensions of graphic grob.

## 36.6.2 Preparing the Display

Add	r. Name	Description
25EE	74 RECLAIMDISP	$(\  ightarrow\ )$
		Activates the text grob, clears it and sets the de-
		fault size.
2EE7	D ClrDAlIsStat	$( \rightarrow )$
		Suspends clock display.
2EEE	D MENUOFF?	( $ ightarrow$ flag )
		Returns TRUE if the menu grob is off.
2F03	34 TURNMENUOFF	$( \rightarrow )$
		Turns off menu display, enlarges ABUFF to fill
		screen.
2F03	31 TURNMENUON	$( \rightarrow )$
		Turns menu grob on.
2EEE	C MENUOFF	$( \rightarrow )$
2624	17 GetHeader	( $\rightarrow$ $\sharp$ )
		Gets header size in lines (0-2).
2628	33 SetHeader	$( \# \rightarrow )$
		Sets header size in lines (0-2).
2609	9 HEIGHTENGROB	( grob #rows $ ightarrow$ )
		Heightens graph or text grob.
260 <i>I</i>	A3 KILLGDISP	$( \rightarrow )$
		Clears graph display by setting it to NULLGROB.
		See DOERASE.
2EEE	79 DOERASE	$( \rightarrow )$
		Erases the graphics display grob without chang-
		ing its size.

## 36.6.3 Immediate Refresh

Addr.	Name	Description
2EF67	SysDisplay	$(\  ightarrow\ )$
		Redisplays all required areas. Does it imme-
		diately, without waiting for the current com-
		mand to finish.
2F19F	?DispCommandLine	( $\rightarrow$ )
		Redisplays the command line now if necessary.
2F19E	DispCommandLine	$( \rightarrow )$
		Redisplays the command line now.
2EE5A	DispEditLine	$( \rightarrow )$
		Just calls DispCommandLine.
2DFCC	?DispMenu	( $\rightarrow$ )
		Redisplays the menu now if no key is waiting
		in the buffer. Even better is this:
		:: DA3OK?NOTIT ?DispMenu ;
2DFF4	DispMenu.1	$(\  ightarrow\ )$
	_	Displays menu now.
2DFE0	DispMenu	$( \rightarrow )$
	-	:: DispMenu.1 SetDAsValid;
2C341	?DispStack	$( \rightarrow )$
	-	Redisplays the stack now if necessary.
2C311	?DispStatus	$( \rightarrow )$
		Redisplays the status area now if necessary.
2C305	DispStatus	$(\rightarrow)$
	2127200000	Displays the status area now.
2C2F9	DispStsBound	$(\rightarrow)$
2021 )	Disposibleana	Displays a horizontal line at y=14, normally
		the separation between header and stack.
2A7F7	DispTimeReq?	( → flag )
211/1	Dispirmenceq.	Is time display required? Checks system flag
		40 and something else.
2F300	DispILPrompt	$(\rightarrow)$
21.300	DISTILL OUR	Redisplays the InputLine prompt, i.e. re-
		freshes the region between the command line
		and the header during InputLine. Requires
		<u> </u>
		a string (the prompt) in 4LAM.

Addr.	Name	Description
26260	nDISPSTACK	( $prompt #height #header flag flag \rightarrow )$
		Used by DispILPrompt.

## 36.6.4 Controlling Display Refresh

Addr.	Name	Description
2EE8D	ClrDA10K	$( \rightarrow )$
2EE8E	ClrDA2aOK	$( \rightarrow )$
2EE8F	ClrDA2bOK	$( \rightarrow )$
2EE90	ClrDA2OK	$( \rightarrow )$
2EE6E	ClrDA30K	$( \rightarrow )$
2EE6D	ClrDAsOK	$( \rightarrow )$
2EE62	DA1OK?	( $\rightarrow$ flag )
2EE63	DA3OK?	( $ ightarrow$ flag )
2EE66	DA2aLess10K?	( $ ightarrow$ flag )
2BF3A	DA1OK?NOTIT	$( \rightarrow )$
		Does DA1OK?, NOT then IT.
2BF53	DA2aOK?NOTIT	$( \rightarrow )$
		DA2aOK?, NOT then IT.
2BF6C	DA2bOK?NOTIT	` '
		DA2bOK?, NOT then IT.
2BF85	DA3OK?NOTIT	$( \rightarrow )$
		Does DA3OK?, NOT then IT.
2EE69	SetDA1Temp	
2EE8A	<u>-</u>	
	SetDA2bTemp	
2EEA7	ClrDA2bTemp	
2F37A	SetDA20KTemp	$( \rightarrow )$
2EE6B	<u>-</u>	
2EE71	<u>-</u>	
2EE64	<u>-</u>	
2EEA5	<u>-</u>	
	SetDA1Valid	
2EF98		
2EE68		` '
	SetDA2Valid	
2EF99	SetDA3Valid	$(\  ightarrow\ )$

Addr.	Name	Description
2EEA0	SetDA3ValidF	$( \rightarrow )$
2EE78	SetDA1Bad	$( \rightarrow )$
2EE74	ClrDA1Bad	$( \rightarrow )$
2EEB0	DA1Bad?	$(\rightarrow flag)$
2EE79	SetDA2aBad	$( \rightarrow )$
2EE75	ClrDA2aBad	$( \rightarrow )$
2EEB1	DA2aBad?	$(\rightarrow flag)$
2EE7A	SetDA2bBad	$( \rightarrow )$
2EEB3	ClrDA2bBad	$( \rightarrow )$
2EEB2	DA2bBad?	$(\rightarrow flag)$
2EE7B	SetDA3Bad	$( \rightarrow )$
2EEB5	ClrDA3Bad	$( \rightarrow )$
2EEB4	DA3Bad?	$(\rightarrow flag)$
2EE72	SetDA1NoCh	$( \rightarrow )$
2EE73	SetDA2aNoCh	$( \rightarrow )$
	SetDA2bNoCh	$( \rightarrow )$
2EE81	ClrDA2bNoCh	$( \rightarrow )$
2EEB7	DA2bNoCh?	$(\rightarrow flag)$
2EE93	SetDA2NoCh	$( \rightarrow )$
2EE6F	SetDA12NoCh	$( \rightarrow )$
2EE77	SetDA3NoCh	$( \rightarrow )$
2EE70	SetDA13NoCh	$( \rightarrow )$
2EE94	SetDA23NoCh	$( \rightarrow )$
2EE65	SetDA12a3NCh	$( \rightarrow )$
		aka: SetDA12a3NoCh
2F379	SetDA123NoCh	$( \rightarrow )$
2EE7C	SetDAsNoCh	$( \rightarrow )$
2EE6C	SetDA2aEcho	$( \rightarrow )$
2EEAC	SetDA1IsStat	$( \rightarrow )$
2EEAE	SetNoRollDA2	
2EEAF	ClrNoRollDA2	
2EEAB	DA1IsStatus?	
2EE7F	SetDA2bIsEdL	
2EE7E	DA2bIsEdL?	$(\rightarrow flag)$
2EE80	ClrDA2bIsEdL	$( \rightarrow )$

## 36.6.5 Clearing the Display

Addr.	Name	Description
25E7E	BLANKIT	( $\#$ startrow $\#$ rows $ o$ )
		Clears #rows from HARDBUFF, starting at
		#startrow.
26021	CLEARVDISP	$( \rightarrow )$
		Clears HARDBUFF.
2EED4	Clr8	$( \rightarrow )$
		Clears top eight rows (first status line).
2EED5	Clr8-15	$( \rightarrow )$
		Clears 2nd status line.
2F15E	Clr16	$( \rightarrow )$
		Clears top 16 rows.
2EF5E	BlankDA1	$( \rightarrow )$
		Clears status area from HARDBUFF.
2F31C	BlankDA2a	$( \rightarrow )$
		Clears display area DA2a.
2F31B	BlankDA2	$( \rightarrow )$
		Clears display areas DA2a and DA2b.
2EE5C	BlankDA12	$( \rightarrow )$
		Clears display areas DA1 and DA2
261C0	CLCD10	$( \rightarrow )$
		Clears status and stack areas.
261C5	CLEARLCD	$( \rightarrow )$
		Clears whole display.
2EF05	DOCLLCD	$( \rightarrow )$
		Like user word CLLCD.

### 36.6.6 Annunciator and Modes Control

Addr.	Name	Description
2613E	SetLeftAnn	$( \rightarrow )$
		Sets left-shift annunciator.
2603A	ClrLeftAnn	$( \rightarrow )$
		Clears left-shift annunciator.
26148	SetRightAnn	$( \rightarrow )$
		Sets right-shift annunciator.

Addr.	Name	Description
2603F	ClrRightAnn	$( \rightarrow )$
		Clears right-shift annunciator.
26139	SetAlphaAnn	$( \rightarrow )$
		Sets alpha annunciator.
26035	ClrAlphaAnn	$( \rightarrow )$
		Clears alpha annunciator.
25EE9	LockAlpha	$( \rightarrow )$
		Sets alpha mode, annunciators, etc.
25F08	UnLockAlpha	$( \rightarrow )$
		Clears alpha mode, annunciators, etc.
2649F	(ClrBusyAnn)	$( \rightarrow )$
		Clears the busy annunciator.
26143	SetPrgmEntry	$( \rightarrow )$
		Sets program-entry mode.
2610C	PrgmEntry?	$(\rightarrow flag)$
		Is program-entry mode set?
25EBE	Do1st/2nd+:	( $\rightarrow$ :: $<$ ob1 $>$ ; (PRG mode) )
		( $\rightarrow$ :: <ob2> <rest> ; (no PRG mode)</rest></ob2>
		)
		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.
25719	SetAlgEntry	$(\rightarrow)$
		Sets algebraic-entry mode.
2571E	ClrAlgEntry	$(\rightarrow)$
		Clears algebraic-entry mode.
256EA	AlgEntry?	$(\rightarrow flag)$
		Is algebraic-entry mode set?
25EDF	ImmedEntry?	$(\rightarrow flag)$
		Returns TRUE if immediate-entry mode (pro-
		gram and algebraic-entry modes cleared).
25E74	?ClrAlg	$(\rightarrow)$
		Clears AlgEntry mode if set.
25E75	?ClrAlgSetPr	$(\rightarrow)$
		Clears AlgEntry mode if set and sets Progra-
		mEntry mode.

## 36.6.7 Window Coordinates

Addr.	Name	Description
2F384	TOP8	( $\rightarrow$ HBgrob #x1 #y #x1+131 #y1+8 )
		Returns coordinates of first status line.
2F36C	Rows8-15	( $\rightarrow$ HBgrob #x1 #y1+8 #x1+131 #y1+16 )
		Returns coordinates of second status line.
2F383	TOP16	( $\rightarrow$ HBgrob $\#$ x1 $\#$ y1 $\#$ x1+131 $\#$ y1+16 )
		Returns coordinates of status area.
2617F	WINDOWCORNER	$( \rightarrow \#x \#y )$
		Gets coordinates of corner of window.
2EED6	HBUFF_X_Y	( $ ightarrow$ HBgrob $\#x$ $\#y$ )
		Returns current grob and window coordinates.
2F352	LEFTCOL	$( \rightarrow \#x )$
		Gets x-coordinate of left column.
2F36B	RIGHTCOL	$( \rightarrow \#x )$
		Gets x-coordinate of right column.
2F385	TOPROW	$( \rightarrow \#y )$
		Gets y-coordinate of top row.
2F31D	BOTROW	$( \rightarrow \#y )$
		Gets y-coordinate of bottom row.
26198	WINDOWXY	$( \#x \#y \rightarrow )$
		Sets corner coordinates.

## 36.6.8 Scrolling the Display

Addr.	Name	Description
26193	WINDOWUP	$( \rightarrow )$
		Moves display one pixel up.
26184	WINDOWDOWN	$( \rightarrow )$
		Moves display one pixel down.
26189	WINDOWLEFT	$( \rightarrow )$
		Moves display one pixel left.
2618E	WINDOWRIGHT	$( \rightarrow )$
		Moves display one pixel right.
2F370	SCROLLUP	$( \rightarrow )$
		Moves display one pixel up, checks for corresponding key being pressed.

Addr.	Name	Description
2F36D	SCROLLDOWN	$(\  ightarrow\ )$
		Moves display one pixel down, checks for corre-
		sponding key being pressed.
2F36E	SCROLLLEFT	$( \rightarrow )$
		Moves display one pixel left, checks for corre-
		sponding key being pressed.
2F36F	SCROLLRIGHT	$( \rightarrow )$
		Moves display one pixel right, checks for corre-
		sponding key being pressed.
2F34A	JUMPTOP	$( \rightarrow )$
		Jumps to top of display.
2F347	JUMPBOT	$( \rightarrow )$
		Jumps to bottom of display.
2F348	JUMPLEFT	$( \rightarrow )$
		Jumps to left of display.
2F349	JUMPRIGHT	$( \rightarrow )$
		Jumps to right of display.
2F38D	WINDOWTOP?	( $ ightarrow$ flag )
		Is window at the top?
2F38A	WINDOWBOT?	( $ ightarrow$ flag )
		Is window at the bottom?
2F38B	WINDOWLEFT?	( $ ightarrow$ flag )
		Is window at the left?
2F38C	WINDOWRIGHT?	( $ ightarrow$ flag )
		Is window at the right?

## 36.6.9 Displaying Objects

Addr.	Name	Description
2F21D	ViewObject	( ob $\rightarrow$ )
2F21E	ViewStrObject	( flag $\$ \to \mathtt{F}$ )
2F21F	ViewGrobObject	Flag decides if it should be possible to toggle TEXT/GRAPH.  ( flag grob → F )
		Flag decides if it should be possible to toggle TEXT/GRAPH.

Addr.	Name	Description
25F12	sstDISP	( ob $\rightarrow$ )
		Displays ob in status line. Used for single
		stepping during debugging.
0C1007	^SCROLLext	( grob $ ightarrow$ )
		Launches PICT environment.
2EF61	WINDOW#	( $\#x \#y \rightarrow$ )
		Internal PVIEW, displays PICT starting at the given coordinates.

## 36.6.10 Displaying Text

Name	Description
DODISP	( ob %row $\rightarrow$ )
	Displays any object in specified row.
DISPROW1	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
	aka: DISP@01, BIGDISPROW1
DISPROW1*	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
	Displays relative to window corner.
DISPROW2	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
	aka: DISP@09, BIGDISPROW2
DISPROW2*	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
	Displays relative to window corner.
DISPROW3	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
	aka: DISP@17, BIGDISPROW3
DISPROW4	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
	aka: DISP@25, BIGDISPROW4
DISPROW5	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
DISPROW6	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
DISPROW7	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
DISPROW8	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
	May not be possible depending on the size of
	the font and whether the menu is on or off.
DISPROW9	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
	May not be possible depending on the size of
	the font and whether the menu is on or off.
DISPROW10	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
	May not be possible depending on the size of
	the font and whether the menu is on or off.
	DODISP  DISPROW1  DISPROW1*  DISPROW2  DISPROW2*  DISPROW3  DISPROW4  DISPROW5  DISPROW6  DISPROW7  DISPROW8  DISPROW9

Addr.	Name	Description
25FB3	DISPN	( \$ #row → )
		aka: BIGDISPN
25EBC	Disp5x7	( $\$$ #start #max $\rightarrow$ )
		Displays string on multiple lines, starting at #start and no using more than #max rows.
		New lines must be manually specified. Seg-
		ments longer than 22 characters are trun-
		cated and appended with "".
25EAD	DISPSTATUS2	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
		Displays message in status area using two
		lines.
38C00	(DISPST2&FREEZE)	$( \ \ \ \ \ \ \ \ \ \ \ )$
		DISPSTATUS2 and freeze status area.
2EEFF	DispCoordl	$( \ \ \ \ \ \ \ \ )$
		Displays \$ in menu grob using minifont.
2F32B	DISPCOORD2	$(\ \ \ \ \ \ \ \ \ )$
		Displays \$ in menu grob using minifont and
		waits for a key. Then refreshes menu display.
25FE5	DISPLASTROW	
		Displays \$ in the last stack display row, just
05		above the menu.
25FEA	DISPLASTROWBUT1	$($\Rightarrow)$
		Displays \$ in the last stack display row. If
25ED4	FlashMsg	menu is turned on it can cover displayed text. ( $\$ \rightarrow$ )
23ED4	riasimsg	Displays message in status area, then re-
		stores it to normal.
2EE61	FlashWarning	( $\$ \rightarrow$ )
20001	Tabliwalliling	Displays message in a message box and beeps.
		Waits for OK to be pressed.
2F1A5	AskQuestion	( $\$ \rightarrow \text{flag}$ )
	~	Use the string to aks the user a question with
		yes/no in a choose box.
02E002	^DoAlert	( \$ → )
		Displays alert messagebox.
2EE60	DoWarning	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
		Displays message, beeps and freezes status
		area.

Addr.	Name	Description
007002	^Ck&DoMsgBox	( $\$$ #x #y grob menu $\to$ T )
		Displays a message box with a grob in the up-
		per left corner and the specified menu. The
		meaning of #x and #y is unclear.
0040B1	~MsgBoxMenu	$( \rightarrow \{ \} )$
		The messsage box menu, with just the OK key.

## 36.6.11 Fonts

Addr.	Name	Description
2621A	FONT>	( $\rightarrow$ font )
		Recalls system font.
2625B	MINIFONT>	( $ ightarrow$ minifont )
		Recalls the current minifont.
25F15	>FONT	( font $\rightarrow$ )
		Sets system font.
2620B	>MINIFONT	( minifont $ ightarrow$ )
		Sets the current minifont.
26288	StackLineHeight	( $\rightarrow$ $\sharp$ )
		Returns height of text grob minus size of
		header and menu.
26242	GetFontStkHeight	$( \rightarrow \# )$
		Returns stack font height (used for display
		stack rows).
		aka: StackFontHeight
25F15 2620B 26288	>FONT >MINIFONT StackLineHeight	<pre>( → minifont ) Recalls the current minifont. ( font → ) Sets system font. ( minifont → ) Sets the current minifont. ( → # ) Returns height of text grob minus size header and menu. ( → # ) Returns stack font height (used for displayment).</pre>

## **Chapter 37 The Menu**

The menu line is divided in six parts, one for each key, each eight pixels high and 21 pixels wide. The starting columns for each menu key label in HARBDUFF2 are:

Hex	$\mathbf{Dec}$	Softkey	Hex	$\mathbf{Dec}$	Softkey
0	0	First softkey (F1)	42	66	Fourth softkey (F4)
16	22	Second softkey (F2)	58	88	Fifth softkey (F5)
2C	44	Third softkey (F3)	$6\mathrm{E}$	110	Sixth softkey (F6)

The command DispMenu.1 redisplays the current menu; and the command DispMenu redisplays the current menu and then calls SetDA3Valid to freeze the menu display area (display area 3).

The words below convert several kinds of objects to menu labels and display them at the specified column:

Word	Stack and action
Str>Menu	( #col \$ → )
	Makes and displays a standard menu label.
Id>Menu	$(\ \texttt{\#col} \ \ \texttt{id} \ \rightarrow)$
	Recalls id and displays standard or directory label, depending
	on the contents.
Grob>Menu	( $\#$ col grob $\rightarrow$ )
	Displays a grob as a menu label.
Seco>Menu	$(\ \texttt{\#col} \ :: \ \rightarrow)$
	Evaluates secondary and uses results to create and display appropriate menu label.

The words below convert strings to the four different kinds of grobs available. All of them take a string and return a grob as arguments

37.1. Menu Format 285

Word	Action
MakeStdLabel	Makes a black label (standard).
MakeBoxLabel	Makes label with a box inside.
MakeDirLabel	Makes directory label (bar above).
MakeInvLabel	Makes white label (like in Solver).

#### 37.1 Menu Format

```
A menu is either a list
{ MenuKey1 MenuKey2 ... MenuKeyN }
or a program
:: <Settings> { MenuKey1 MenuKey2 ... MenuKeyN } ;
which returns such a list and optionally changes of the default menu properties installed by InitMenu.
```

Each menu key can be any of the following:

```
NullMenuKey
KeyObj
{ LabelObj KeyProcNS }
{ LabelObj { KeyProcNS KeyProcLS } }
{ LabelObj { KeyProcNS KeyProcLS KeyProcRS } }
```

LabelObj is the object to be displayed as the label. If it is a program with TakeOver as the first command, it is evaluated with the x-position of the label on the stack and must return the argument(s) for the LabelDef program (normally the x-position of the label and the object to display as a label).

If you do not override the LabelDef command (most of the times you will not), then LabelObj can be any object, but genereally it is a string or a 21x8 grob.

KeyProc is the action taken upon key press. It will be executed by a special executor which takes appropriate actions depending upon the object type. If KeyProc is a program with TakeOver as the first command, it will override the normal executor. NS here means this is the action when the menu key is pressed unshifted (think of No-Shift). Similarly, LS and RS means the actions run when the key is pressed left- or right-shifted, respectively.

286 37. The Menu

### 37.2 Menu Properties

The menu system of the HP49 provides an amazing flexibility. Besides the normal actions, a menu has many properties which define the appearance of labels and the specific actions taken upon keypresses, actions to take when the context changes or a different menu is installed etc.

The properties a menu carries are:

Word	Stack and action	
MenuDef	The current menu.	
MenuKeys	The menu keys in a list.	
MenuRow	The menu page.	
LabelDef	The label builder for menu.	
MenuRowAct	Action taken when menu row changes or when LastMenu is	
	reinstalled.	
ExitAction	Action taken when menu changes. Normally this action	
	saves the current menu as LastMenu.	
TrackAct	Action taken when the context (the current directory)	
	changes.	
ReviewKey	Action taken when REVIEW key (Rightshift DOWN) is	
	pressed.	
MenuKeysNS	Action taken when menu key is pressed.	
MenuKeysLS	Action taken when menu key is pressed left-shifted.	
MenuKeysRS	Action taken when menu key is pressed right-shifted.	
BadMenu?	Must the menu be be redrawn?	
Rebuild?	Has the menu row changed?	
Track?	If context has changed is there a prg to execute?	

Examples for the TrackAct property are:

- SolverMenu has DoSolveMenu as TrackAct, because there might be another EQ variable to use.
- The Custom menu just restarts itself because the value of the CST variable may have changed. (CstTrack = :: NoExitAction MenuDef@ InitMenu;)

Most menu properties can be modified using supported entry points. Here is an example for doing so. The following program sets a modified VAR menu, which allows variables to be protected against being overwritten with a left-shifted menukey.

```
1
   ::
      MenuMaker
      ::
        ROMPTR A9 2
                                         (the builtin VAR menu)
5
        ::
          DUP
          DUPTYPECSTR? NOT_IT DECOMP$
                                         (make a string)
          SWAP
10
          ID prtct
          ITE
                                          (select label type)
            MakeInvLabel
            MakeStdLabel
          Grob>Menu
                                          (display label)
15
        LabelDef!
         ::
          ID prtct
20
          NOTcase xSTO
          DECOMP$ "\OA is protected" &$
          FlashWarning
        MenuKeyLS!
25
      DoMenuKey
```

This does several things:

- 1. Gets the normal VAR menu in order to pass it to DoMenuKey.
- 2. It modifies the LabelDef property in a way that the protected variables will have an inverted label in the menu.
- 3. It modifies the MenuKeyLS property in a way that it exits with an error message if the relevant variable is protected.

The program in ID prtct is an ID selector which has the stack diagram (  $id \rightarrow flag$  ) and must decide if a given ID should be protected. Here are some possibilities:

1. :: DROPTRUE ; — All variables are protected

288 37. The Menu

- 2. :: DROPFALSE ; No variables are protected
- 3. :: ID>\$ CAR\$ CHR\_% EQUAL ; Variables with names starting with "%" are protected.

Note that if the variable prtct does not exist or does not follow the required stack diagram, the calculator may crash. You might want to modify the program to put in better protection against user errors. Also note that this only protects against storing using left-shift and a menukey. It will not protect against using the STO command or the filer, naturally.

#### 37.3 Reference

#### 37.3.1 Menu Properties

Name	Description
GETDF	( $\#$ menukey $\rightarrow$ ob )
	Gets the definition of a menu key from
	THOUCHTAB. $\#$ menukey = $\#1\#6$
GETPROC	( $\#$ menukey $ o$ ob )
	Gets the definition of a menu key from
	THOUCHTAB. #menukey = #1#6. With #7, get
	the executor.
SetRebuild	$( \rightarrow )$
	Sets the flag that the menu needs to be rebuild.
MenuRow!	( $\#n \rightarrow$ )
	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,
MenuRow@	( $\rightarrow$ #n )
	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.
	GETDF  GETPROC  SetRebuild  MenuRow!

Addr.	Name	Description
260A8	LastMenuRow!	( $\#n \rightarrow$ ) 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. $1,7,13,$
260AD	LastMenuRow@	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.
25845	MenuDef@	( → menu )  Recalls the current menu definition. menu is a  MenuList or a program, or a Rompointer.
25908	LastMenuDef!	<ul> <li>( menu → )</li> <li>Sets the definition of the last menu. menu is a</li> <li>MenuList or a program, or a Rompointer.</li> </ul>
2590D	LastMenuDef@	( → menu )  Recalls the definition of the last menu. menu is a MenuList or a program, or a Rompointer.
25EFB	SaveLastMenu	( $\rightarrow$ ) Stores row and definition of current menu as the last menu.
25EDA	GetMenu%	$( \rightarrow \% )$
25863	MenuRowAct!	( ob $\rightarrow$ ) Stores ob as the RowAct menu property.
25EE2	InitTrack:	( $\rightarrow$ ) Execute the program which is next in the runstream if the directory changes. Used by the VAR menu to set first menurow when directory changes, or by the CST menu to rebuild it.

290 37. The Menu

Addr.	Name	Description
25877	LabelDef!	( ob → )  Store a program which displays a menu label.  Prg has the stack diagram  ( #col ob → )  For example, the LIBS command uses the following program to make all menu label look like directories:  :: DUPNULL\$? ITE  MakeStdLabel MakeDirLabel  Grob>Menu;
		During execution, INDEX@ will contain the menu key number.
2589F	MenuKeyLS!	( ob $\rightarrow$ 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}.
258B3	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}.
2588B	MenuKeyNS!	(og $\rightarrow$ 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}.
25890	MenuKeyNS@	$(\rightarrow \text{ ob })$ Recall the action for unshifted menu keys.
25EFC	SetKeysNS	( ob $\rightarrow$ ) Sets ob as MenuKeysNS, DoBadKey to LS & RS.
25F02	StdMenuKeyLS	( $\{\text{ob-NS ob-LS ob-RS}\} \rightarrow ?$ ) The content of MenuKeyLS for standard menus.
25F03	StdMenuKeyNS	( ob-NS $\rightarrow$ ? ) ( {ob-NS ob-LS ob-RS} $\rightarrow$ ? ) The content of MenuKeyNS for standard menus.

Addr.	Name	Description
27FED	NullMenuKey	$(\  ightarrow\ )$
		A placeholder for an empty menu key when
		defining menu lists.
258C7	ReviewKey!	( ob $\rightarrow$ )
		Store a program which is called with the review
		key (RS DOWN). The program has the stack di-
		agram
		$( \rightarrow )$
258EF	(ExitAction!)	( ob $\rightarrow$ )
		Store ob as exit action.
25EEF	NoExitAction	$( \rightarrow )$
		Sets NOP as ExitAction. Mostly used to avoid
		that the menu is saved as the previous menu
		when a new Menu gets installed.

## 37.3.2 Building Menus

Addr.	Name	Description
275C6	TakeOver	$(\  ightarrow\ )$
		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 pro-
		gram in the label slot of a menu key, the pro-
		gram is evaluated to get the label object (most
07500	Modifier	likely a grob). ( $\rightarrow$ )
275EE	Modifier	( → ) :: TakeOver ;
27620	MenuMaker	$( \rightarrow ob )$
27020	menumaker	Quotes next object, and also provides
		TakeOver. The disassembly is
		:: TakeOver 'R ;
		Normally this is used like this:
		:: MenuMaker menu InitMenu ;
25EE0	InitMenu	( $menu \rightarrow$ )
		menu is {} or :: settings {} ; Settings override
		the default settings installed by InitMenu.
25EC6	DoMenuKey	$($ menu $\rightarrow$ $)$
		:: SetDA12NoCh InitMenu ;

292 37. The Menu

Addr.	Name	Description
25EE1	InitMenu%	( $%mnu.pg \rightarrow )$
		( %0 → )
25F00	StartMenu	( menu $\#n \rightarrow$ )
		#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.
25EFE	SetThisRow	$( \rightarrow )$
		Builds a new TOUCHTAB, SetBadMenu.
25EE8	LoadTouchTbl	( MenuKey1 MenuKeyN $\#$ n $ o$ )
		Builds new TOUCHTAB from menukeys.

## 37.3.3 Menu Display

Addr.	Name	Description
2EF66	SysMenuCheck	$( \rightarrow )$
		Checks menu validity. If DA3NoCh? then noth-
		ing. If Track? then ?DoTrackAct@. If Rebuild?
		then SetThisRow.
2DFCC	?DispMenu	$( \rightarrow )$
		Redisplay the menu now if no key is waiting in
		the buffer. Even better is this:
		:: DA3OK?NOTIT ?DispMenu ;
2DFF4	DispMenu.1	$( \rightarrow )$
		Displays the menu immediately.
2DFE0	DispMenu	$( \rightarrow )$
		:: DispMenu.1 SetDAsValid;

## 37.3.4 Displaying Menu Labels

Addr.	Name	Description
2E0D5	Grob>Menu	( $\#$ col grob $\rightarrow$ )
		Displays grob as menu label.
2E0F3	Str>Menu	( $\#col $ \rightarrow )$
		Displays string as menu label.
2E11B	Id>Menu	( $\#$ col id $\to$ )
		Displays id as menu label.
2E107	Seco>Menu	( $\#$ col :: $\rightarrow$ )
		Does EVAL then DoLabel.
25886	DoLabel	( $\#$ col ob $\to$ )
		If ob is of one of the supported types, displays a
		menu label. If not, generates a "Bad Argument
		Type" error.
2E2AA	MakeLabel	( $\$$ #w #x grob $\rightarrow$ grob' )
		Inserts \$ into grob using CENTER\$3x5 with y=5.
08E007	^WRITEMENU	$(\$6\$1 \rightarrow )$
		Displays the six strings as menu keys.

#### 37.3.5 General Entries

Addr.	Name	Description
25EA6	CheckMenuRow	( # → # #' )
25EFD	SetSomeRow	$( \#n \rightarrow )$
		with $Mod(n,FFFFFFh) = 0$ .
2589A	DoMenuKeyNS	$( \#n \rightarrow )$
275FD	MenuKey	$( \rightarrow )$
		Takes NOB from Runstream.
2F15B	CLEARMENU	$( \rightarrow )$
25F2B	CHECKMENU	$( \rightarrow )$
3EA01	(CST)	( $ ightarrow$ ob )
		Evaluates ID CST.
2C2C0	nCustomMenu	$( \rightarrow )$
		Installs the CST menu.
25EFF	SolvMenuInit	$( \rightarrow )$
		Sets MenuKeyNS/LS/RS, ReviewKey and La-
		belDef properties needed by the Solver menu.

294 37. The Menu

Addr.	Name	Description
25EC3	DoFirstRow	$( \rightarrow )$
		Sets the first row of the current menu

# **Chapter 38**Programming the HP49 Editor

The HP49G has a builtin editor which is much faster and nicer than the editor on the HP48. However, it is a general-purpose editor, and it would be useful for specific applications to add some features without having to write a whole new editor. The HP49G ROM contains a number of supported entry points which can be used to manipulate the editor from programs. These can be used to write editor extensions.

## 38.1 Terminology

The terms below will appear often in this chapter.

Term	Meaning
EditLine	The string which is currently being edited. Also called
	"Buffer" and "Command line". In the stack diagrams, we
	will use \$buf for it.
Cursor position	The position of the cursor in the Editline. Represented by
	a bint. In stack diagrams, is written as #cpos.
Current line	The current line in the editor, i.e. the substring after the
	NEWLINE before the cursor up to the next NEWLINE
	character.
Editor window	When the text being edited is too long and/or wide, the
	screen of the HP49G shows only a part of the text: the
	window. When the cursor is moved, the window must be
	re-positioned to show the new position.
Selection	A region in the buffer can be selected when the begin
	marker and the end marker are active. The selected sub-
	string is called \$sel in the stack diagrams.

Term	Meaning
Word-start	The beginning of a word, a position in a string where the
	char before is SPACE or NEWLINE, and the char after is
	a non-white character. Several commands deal with word-
	start positions, called #ws in the stack diagrams below.
Invisible chars	The HP49G can show text in different fonts and styles. In
	order to switch between fonts and styles, special markers
	are inserted into the text to indicates a change in font or
	style. These 3-character sequences are not visible, but they
	count in string length and in cursor position. Some Editor
	commands are aware of these strings and do complicated
	computations to cut and paste text with attributes. This of
	course makes these commands slower than they could be.
	If you do not use fonts and styles, you need not to worry
	about all this.

## 38.2 Examples

For information on the specific entries used in the examples below, consult the Reference section below.

1. Select the current line and copy it onto the clipboard.

```
1
    ::
      TakeOver
      CMD_END_LINE
                           (goto end of line)
      RCL CMD POS
                           (recall position)
5
      CMD_STO_FIN
                           (store as marker)
      CMD_DEB_LINE
                           (beginning of line)
                           (recall position)
      RCL_CMD_POS
      CMD_STO_DEBUT
                           (store as marker)
      CMD_COPY
                           (copy to clipboard)
10
```

This can be done shorter by using the builtin command SELECT.LINE command. The following is equivalent to the above.

```
1 :: TakeOver
```

```
SELECT.LINE CMD_COPY 5 ;
```

2. Insert a ":: ;" template on a single line and position the cursor between "::" and ";".

3. Insert a multi-line ":: ;" template and position the cursor with extra indentation on the second line.

4. Go to next label. Labels are lines starting with "\*".

```
1 ::
      TakeOver
       "\0A*"
                           (newline followed by star)
      {\tt FindStrInCmd}
                           (find that)
 5
       ΙT
         ::
                           (if successful)
           DROP
                           (drop #end)
                           (correct to move over NL)
           #1+
           STO_CURS_POS
                           (set new cursor position)
10
         ;
      DROP
                           (drop the search string)
```

5. The RPLCPL command of the Emacs library (see section A.6) does completion of names in the Editor. It needs to find the word fragment before the cursor. Here is how this can be done:

```
1
   ::
                    (recall EditLine)
     RCL CMD
                    (current position)
     RCL_CMD_POS
                    (arg needed by GET.W<-)
     DUP
5
                    (position of word start)
     GET.W<-
                    (prepare args for SUB$)
     #1+SWAP
     SUB$
                    (get the substring)
   ;
```

6. Change the indentation of the current line to #N spaces. #N is a bint expected on stack level 1. The command leaves empty lines and lines starting with a "\*" alone.

```
1
   ::
      Blank$
                         (make the indentation str.)
      CMD_DEB_LINE
                         (goto beginning of line)
      RCL_CMD
5
      RCL_CMD_POS
      #1+ SUB$1
                         (look at first char in line)
                         (ASCII code of '*')
      BINT42
      OVER#=case
                         (line starts with '*'?)
         :: 2DROP
                         (cleanup,
10
                         ( next line & exit)
           CMD_DOWN
      BINT32 >#?SKIP
                         (line starts with nonwhite ch)
        ::
          CMD_END_LINE
                         (line starts with whitespace:)
15
          RCL_CMD_POS
                         (remember end of line position)
                         (back to beginning of line)
          CMD_DEB_LINE
          DO>Skip
                         (jump to next word)
          RCL_CMD_POS
          #<ITE
20
            DROPRDROP
                         (if already in next line: Exit)
                         (kill whitespc before 1st word)
            DoFarBS
      CMD PLUS
                         (insert spaces)
      CMD DEB LINE
                         (back to beginning of line)
25
      CMD DOWN
                         (next line)
```

## 38.3 Executing External Commands in the Editor

In order to use the new commands in the editor, you must bind them to a key or put them into a menu. Note that each command you write needs a TakeOver as the first entry in the secondary or the command will not execute in the editor.

Here is a simple example for an InputLine environment which defines an initial menu with two commands to select the current line and to clear the EditLine. For more information on InputLine, see Chapter 31.

```
1
       "Edit this!"
                           (prompt)
                           (initial string)
                           (cursor position)
5
                           (modes)
      zerozerozero
           "SLINE"
           ::
                           (program to select line)
10
             TakeOver
             SELECT.LINE
           "CLEAR"
15
                           (program to clear EditLine)
           ::
             TakeOver
             DEL_CMD
20
                           (initial menu line)
      ONE
      TRUE
                           (abort flag)
                           (parse)
      ZERO
25
      InputLine
                           (and GO!)
```

## 38.4 Reference

## 38.4.1 Status

Addr.	Name	Description
257A2	EditLExists?	( $ ightarrow$ flag )
		Does an EditLine exist?
2EEED	NoEditLine?	( $ ightarrow$ flag )
		Does no EditLine exist?
2F196	RCL_CMD	$( \rightarrow \ \ )$
		Returns a copy of the current command line to
		the stack. Same as EDITLINE\$.
2EEEB	EDITLINE\$	$(\  ightarrow\ \ \ \ )$
		Returns a copy of the current command line to
		the stack. Same as RCL_CMD.
2F197	RCL_CMD2	$(\  ightarrow\ \ \ \ )$
		Similar to RCL_CMD, but if there is not enough
		memory to copy the EditLine to the stack, it
		will move the current EditLine into TEMPOB.
		Of course, this will delete the current Edit-
0000	DGI GMD DOG	Line.
2EF87	RCL_CMD_POS	
26505	CLID COD C	Recalls the current cursor position. $( \rightarrow \# )$
26585	CURSOR@	•
26594	(CURSOR PART)	Recalls the current cursor position. ( $\rightarrow$ # )
20394	(CURSUR_PART)	$( \rightarrow \# )$ Recalls the current cursor row (line).
2F158	(THISCHAR)	( $\rightarrow$ chr )
ZF 130	(Inischar)	Returns the character under the cursor. At the
		end of the file, returns CHR_00.
2EEEA	CURSOR_END?	$(\rightarrow flag)$
20007	CORBOR_HID:	Checks if the cursor is at the end of a line or at
		the end of the file. Works by checking the cur-
		rent character against newline and CHR_00.
264CC	FIRSTC@	$(\rightarrow \#)$
<del>-</del>		Column of the left display window edge.
26030	CURSOR_OFF	$(\ \rightarrow\ \sharp\ )$
	<del>-</del>	Cursor column relative to left edge of display
		window.

Addr.	Name	Description
2EF91	CAL_CURS_POS	( #1 #c → # ) Computes a position in the current EditLine from line and column number. The result can be used by STO_CURS_POS to move the cursor to that location. If #line is larger than the number of lines in the EditLine, computes the position of the last line.
2EF90	CAL_CURS_POS_VIS	( $\sharp 1 \ \sharp c \to \sharp$ ) Similar to CAL_CURS_POS, but will ignore invisible characters. The result can be used by STO_CURS_POS_VIS to move the cursor to that location.
2F199	RCL_CMD_MODE	( → \$ ) Recalls a string with current editor settings. Can be used together with STO_CMD_MODE to save and restore the state of the EditLine, when temporarily leaving the editor with HALT or when calling a program which must temporarily change settings
2F198	STO_CMD_MODE	porarily change settings. ( $\$ \to $ ) Stores a mode string similar to the one obtained by RCL_CMD_MODE.

# 38.4.2 Inserting Text

Addr.	Name	Description
2EF74	CMD_PLUS	$( \ \$ \  o \ )$
		Inserts string at current cursor position in Edit-
		Line.
2F194	CMD_PLUS2	$( \ \ \ \ \ \ \ \ \ \ \ )$
		Replaces entire current EditLine with new
		string. When there is not enough memory to
		copy the string on stack level 1, moves the string
		out of TEMPOB. You must be careful that the
		string is not referenced in any way. The cursor
		is moved to the end of the new string.

Addr.	Name	Description
2F195	CMD_PLUS3	( \$ → )
		Same as CMD_PLUS2, but the cursor position is
		not changed. Useful when restoring a command
		line context after HALT.
2EF97	InsertEcho	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
		Inserts string at current cursor position in Edit-
		Line.
2EEE4	Echo\$Key	$( \$/chr \rightarrow )$
		Same as CMD_PLUS.
2F11C	Echo\$NoChr00	$( \ \ \ \ \ \ \ \ \ \ \ )$
		Inserts string at current cursor position in Edit-
		Line.
25EC1	DoDelim	$( \rightarrow )$
		Takes a character or string from the runstream
		and inserts it.
25EC2	DoDelims	$( \rightarrow )$
		Takes a character or a string from the run-
		stream, inserts it and moves the cursor back by
		one character.
25795	INSERT_MODE	$( \rightarrow )$
		Turns insert mode on. In insert mode, new char-
		acters do not overwrite old ones.
2577F	(TogInsert)	$( \rightarrow )$
		Toggles the insert/overwrite flag.
25790	INSERT?	$(\rightarrow flag)$
		Returns TRUE if insert mode is active.

# 38.4.3 Deleting Text

Addr.	Name	Description
2EF82	CMD_DEL	$( \rightarrow )$
		Deletes next char in Editor. Same as LS+DEL. If you hold down BS while this entry is executed, the HP49G will think you have pressed the key and want to repeat it.

Addr.	Name	Description
2EF81	CMD_DROP	( → ) Backspace in Editor. Deletes char before cursor. Same as BS key. If you hold down BS while this entry is executed, the HP49G will think you have pressed the key and want to repeat it.
2EF95	DEL_CMD	$(\  ightarrow\ )$ Clears the entire EditLine.
2EEE7	InitEdLine	$(\rightarrow)$ :: DEL_CMD ;
2F2F0	DO <del< td=""><td>( <math>\rightarrow</math> ) Deletes left to beginning of word. Same as the</td></del<>	( $\rightarrow$ ) Deletes left to beginning of word. Same as the
2F2F1	DO>Del	<ul> <li>←DEL button in the editor TOOL menu.</li> <li>( → )</li> <li>Deletes right to beginning of next word, Same as the DEL→ button in the editor TOOL menu.</li> </ul>
2F2F9	DODEL.L	( $\rightarrow$ ) Deletes all chars in the current line. If the line is already empty, delete the NEWLINE. Same as the DEL.L button in the editor TOOL menu.
2F2DD	DoFarBS	$(\rightarrow)$ Deletes to beginning of line. Same as the RS+ $\leftarrow$ DEL in the editor TOOL menu.
2F2DE	DoFarDel	( $\rightarrow$ ) Deletes to end of line. Same as RS+Del $\rightarrow$ in the editor TOOL menu.

# 38.4.4 Moving the Cursor

Addr.	Name	Description
2EF8B	STO_CURS_POS	( # → ) Stores cursor position. Moves cursor to specified position and if necessary repositions the editor window to make sure the cursor position is visible. If it is necessary to scroll the window horizontally, this command sets the left edge of the window to the cursor column and shows as much text as possible to the right of the cursor. However, if the cursor is also visible when the window edge is moved to column zero, this position takes precedence.
2EF8C	STO_CURS_POS2	( $\# \to$ ) Same as STO_CURS_POS, but moves the right edge of the editor window to the cursor column.
2EF8D	STO_CURS_POS3	( ♯ → ) Same as STO_CURS_POS, but without checking for style/font switch sequences. So while STO_CURS_POS always makes sure the cursor ends up right before a visible character, this command allows you to position it within the invisible escape sequences.
2EF8E	STO_CURS_POS4	( # → ) Behaves with respect to editor window positioning like STO_CURS_POS2, but with respect to invisible chars like STO_CURS_POS3.
2EF8F	STO_CURS_POS_VIS	( $\#\to$ ) Like STO_CURS_POS, but ignores the invisible characters. So if you look at your string and say, I want to go to what I see as the 5th character, use this entry.
2F378	SetCursor	( $\# \to$ ) ( $\{\# \#'\} \to$ ) Sets the cursor to the given position. For the list argument, the numbers are row and column.

Addr.	Name	Description
2EF7C	CMD_NXT	$(\  ightarrow\ )$
		Moves cursor to next char, like Right Arrow.
2EF7B	CMD_BAK	$( \rightarrow )$
		Moves cursor to the left. Same as as Left Ar-
		row.
2EF80	CMD_DOWN	$(\rightarrow)$
		Moves cursor to the next line. Same as Down
		Arrow.
2EF7F	CMD_UP	$(\rightarrow)$
		Moves cursor to the previous line, like Up Ar-
		row.
2EF7D	CMD_DEB_LINE	$(\rightarrow)$
		Moves cursor to the beginning of line. Same as
		RS+LEFT.
2EF7E	CMD_END_LINE	$(\rightarrow)$
		Moves cursor to the end of line. Same as
0		RS+RIGHT.
2EF7A	CMD_PAGED	$(\rightarrow)$
0		Moves cursor one page down, like LS+DOWN.
2EF77	CMD_PAGEL	$(\hspace{.1cm} \rightarrow \hspace{.1cm})$
00070		Moves cursor one page left, like LS+LEFT.
2EF78	CMD_PAGER	( → )
20070	CMD DAGELL	Moves cursor one page right, like LS+RIGHT. ( $\rightarrow$ )
2EF79	CMD_PAGEU	· · · · · ·
Omonin	DO < Clrin	Moves cursor one page up, like LS+UP. ( $\rightarrow$ )
2F2EE	DO <skip< td=""><td>Skips left to beginning of word. Same as the</td></skip<>	Skips left to beginning of word. Same as the
		←SKIP button in the editor TOOL menu.
2F2EF	DO>Skip	$(\rightarrow)$
	DO\2KID	Skips right to the beginning of the next word.
		Same as the SKIP— button in the editor TOOL
		menu.
2F2E4	DO>BEG	$(\rightarrow)$
21211	20/200	Goes to begin of selection (if active) or to be-
		ginning of EditLine. Same as →BEG button in
		the editor TOOL menu.

Addr.	Name	Description
2F2E5	DO>END	$(\  ightarrow\ )$
		Goes to end of selection. Same as the $\rightarrow$ END
		button in the editor TOOL menu. When there
		is no selection, does not move.
2F2E6	GOTOLABEL	$( \rightarrow )$
		Brings up the CHOOSE-box with labels in the
		EditLine. Same as the LABEL button in the
		editor TOOL/GOTO menu.

### 38.4.5 Selection, Cut and Paste, the Clipboard

Addr.	Name	Description
2EF83	CMD_STO_DEBUT	( ♯ → )
		Sets begin marker, like RS+BEGIN, but takes
		position from stack.
2EF84	CMD_STO_FIN	$( \# \rightarrow )$
		Sets end marker, like RS+END, but takes posi-
		tion from stack.
2EF85	RCL_CMD_DEB	$( \rightarrow \# )$
		$( \rightarrow \#0 )$
		Recalls the position of the BEGIN marker. If
		the selection has been cleared, returns ZERO.
2EF86	RCL_CMD_FIN	$( \rightarrow \# )$
		$( \rightarrow \#0 )$
		Recalls the position of the END marker. If the
		selection has been cleared, returns ZERO.
2F2DC	ClearSelection	$(\  ightarrow\ )$
		Unselects the selected text without changing
		the contents of the editor. Sets both begin and
		end marker to ZERO.
2EF93	VERIF_SELECTION	$(\rightarrow flag)$
		Returns TRUE when the END marker is not
		ZERO, indicating that the selection is active.
		Use this command as a check before doing any-
		thing with the selection.
2EF8A	CMD_COPY	$(\rightarrow)$
		Copies selected string, like RS+COPY.

Addr.	Name	Description
2EF88	CMD_CUT	<ul> <li>( → )</li> <li>Cuts string. Really is "delete", does not copy to kill buffer. So a "normal" CUT would be</li> <li>:: CMD_COPY CMD_CUT ;</li> </ul>
2F2FA	CMD_COPY.SBR	Puts the selection as a string on the stack. This command is font/style aware. It is recommended not to use it because it may get the wrong text style if the cursor is not repositioned to the beginning of the selection first. If you don't use fonts,  :: RCL_CMD  RCL_CMD_DEB RCL_CMD_FIN
2EF94	PASTE.EXT	SUB\$ ; does something similar.  (\$ → )  Pastes from stack with treatment of fonts and styles. Inserts the string on stack level at the cursor position. It can insert normal text right in the middle of bold test etc. If you don't use styles or different fonts, CMD_PLUS is probably faster.
2F2E1	SELECT.LINE	ster.  ( → )  Selects current line, position cursor at beginning of line. Selection does not include the NEWLINE char at the end of the line.
2F2E2	SELECT.LINEEND	$(\rightarrow)$ Selects current line, position cursor at end of line. Selection does not include the NEWLINE char at the end of the line.
2A085	(Clipboard!)	$(\ \ \ \ \ \ \ \ )$ Stores string to Clipboard.
2A095	(Clipboard@)	$( \rightarrow \$ )$ Recalls Clipboard contents to stack.
2A0A5	(Clipboard0)	$(\rightarrow)$ Clears the Clipboard.
2A0B5	(Clipboard?)	<pre>( → flag ) Is there anything on the Clipboard?</pre>

# 38.4.6 Search and Replace

Addr.	Name	Description
2F2F3	GET.W->	( → # ) Returns the position of the next word-start to the right of the current cursor position. Note the asymmetry of this command and
2F2F4	GET.W<-	GET.W< ( $\# \to \#'$ ) Takes a position from the stack and return the position if the nearest word-start to the left of that position. Note the asymmetry of this command and GET.W->.
2F2F2	FindStrInCmd	( \$find → \$find \$start \$end T ) ( \$find → \$find F ) Finds a string in the EditLine, starting from the current cursor position. The search string remains on the stack, presumably in order to do repeated searches. Returns the start and end positions of the match and a flag. This function respects the setting of the internal flag for case-sensitive search.
2F2E8	DOFIND	Same as the FIND menu button in the editor TOOL/SEARCH menu. Pops up the FIND input form.
2F2EA	DONEXT	$( \rightarrow )$ Finds next. Same as the NEXT button in the editor TOOL/SEARCH menu.
2F2E9	DOREPL	( $\rightarrow$ ) Same as the REP button in the editor TOOL/SEARCH menu. Pops up the REPLACE input form.
2F2EB	DOREPLACE	( $\rightarrow$ ) Replaces current match. Same as the R button in the editor TOOL/SEARCH menu.

Addr.	Name	Description
2F2EC	DOREPLACE/NEXT	$( \rightarrow )$
		Replaces current match and move to next
		match. Same as the R/N button in the edi-
		tor TOOL/SEARCH menu.
2F2ED	REPLACEALL	$( \rightarrow )$
		Replaces all matches in buffer. Same as the
		ALL button in the editor TOOL/SEARCH
		menu.
2F2FC	REPLACEALLNOSCREEN	$( \rightarrow )$
		Like REPLACEALL, but does not update the
		screen. Much faster this way.

#### 38.4.7 Evaluation

Addr.	Name	Description
2F2DF	EditSelect	$( \rightarrow )$
		Edits the current selection. Opens the editor
		with the selection only. You can then edit the
		selection. After pressing ENTER the edited
		text is inserted back into the previous editing
		environment.
2F2E3	EVAL.LINE	$( \rightarrow )$
		Evaluates the current line and replace it with the result of the evaluation. Similar to EVAL.SELECTION, but without the need to se-
		lect the line first.
2F2FB	EVAL.SELECTION	$(\  ightarrow\ )$
		Evaluates the current selection and replace it with the result of the evaluation. Same as the EXEC button in the editor TOOL menu.

Addr.	Name	Description
2F2F8	EXEC_CMD	( cmd algflag $ ightarrow$ obsel )
		Runs a command on the selection in the Edit-
		line. Takes two arguments: the command to
		run and a flag which says how to compile the
		selection before the command is applied. If the
		flag is TRUE, and ALG mode in on, the ALG
		compiler is used and the DOTAG :: $xEVAL$
		prologue of the result is removed. Use this if
		the result is to be edited by another editor. The
		selection is left on stack level 1 as an object.
0B954	(RunInNewContext)	( ob $\rightarrow$ )
		Saves current user interface, evaluate ob and restore the user interface. Can be used to run applications from inside another application.

# 38.4.8 Starting the Editor

Addr.	Name	Description
2EEE9	EditString	(\$ → ) Starts editing the string when the current program exits. This is the entry to use if a program should exit with the editor activated. Use InitEdLine before this entry to clear the editline (if desired) - if not, the string is inserted into the current editline. All code after this entry will be executed before control is handed to the editor application. For example:
		"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)  ;

Addr.	Name	Description
2F19A	ViewLevel1	$( ob \rightarrow ob' )$
		Edits the object in level 1
2F1AF	AlgObEdit	$( ob \rightarrow ob' )$
		Used instead of ViewLevell if in Algebraic mode. Does not execute STARTED and EXITED.
2B2F2	(CallEditCmd:)	$( ob \rightarrow ob' )$
		Evaluates the next object in the runstream, which usually in an editing command like ObEdit. When the evaluation returns FALSE, the original object which was saved in a temporary variable is restored to the stack. When the evaluation returns TRUE, the TRUE is removed from the stack.
2EEE5	EditLevel1	$( ob \rightarrow ob' )$
2F1AE	ObEdit	$( ob \rightarrow ob' T )$
		( ob $ ightarrow$ F )
		Edits object. When the user cancels, only FALSE is returned. Otherwise the changed object along with TRUE is returned.
011004	^EQW3Edit	( symb $\rightarrow$ symb' T ) ( symb $\rightarrow$ F ) Opens the equation editor to edit the expression. If exited by ENTER, returns new expression and TRUE. If exited by CANCEL, returns just FALSE.

### 38.4.9 Miscellaneous

Addr.	Name	Description
25ED2	EditMenu	( → {} )
		Returns the Editor menu.
2EF73	?Space/Go>	$( \rightarrow )$
		Inserts a SPACE character unless there is already one before the cursor position. Use this if you want to make sure the next stuff echoed is separated by at least one space from the word preceding it.

Addr.	Name	Description
2EF76	AddLeadingSpace	( \$ → \$' )
		Adds a leading space to the string on level1 if
		it does not start with a space and if the cursor
		in the editor is after a non-white character. So
		:: "DUP" AddLeadingSpace
		AddTrailingSpace CMD_PLUS ;
		inserts DUP and makes sure it will be sur-
		rounded by spaces.
2EF75	AddTrailingSpace	( \$ → \$' )
		Adds a trailing space to the string on level1
		unless the string already ends with a space.
2EF9A	CommandLineHeight	$(\rightarrow \#pix)$
		Returns the number pixel rows occupied by
		visible part of the EditLine.
2F2DB	DOTEXTINFO	$( \rightarrow )$
		Displays the info screen about the Editline.
		Same as the INFO button in the editor TOOL
		menu.
2F2F6	GET_CUR_FONT.EXT	$( \rightarrow \# )$
		Returns the ID (as a system binary) of the
		font used for the character under the cursor.
2EF96	NO_AFFCMD	$( \rightarrow )$
		Tells the next CMD_PLUS call not to update
		the display. For speed, if you want to do more
		insertion before the user needs to see it.
2F19E	DispCommandLine	$( \rightarrow )$
		Redisplays the command line.
2F19F	?DispCommandLine	$( \rightarrow )$
		Redisplays the command line if necessary.

Addr.	Name	Description
2EF92	XLINE_SIZE?	( ob → flag ) Checks if the cursor is outside the current line. In the HP49G editor, you can move the cursor further to the right than the line length, without actually making the line longer. The line gets extended only if you actually insert text or use CMD_DEL to catch to following line to the position. This entry returns TRUE if it is not on or before the newline. Note that it takes an arbitrary object from the stack first - so put something there before calling it.
27F47	<delkey< td=""><td><math>( \rightarrow \{\} )</math> Returns the <math>\leftarrow</math>DEL menu key.</td></delkey<>	$( \rightarrow \{\} )$ Returns the $\leftarrow$ DEL menu key.
27F9A	>DelKey	$(\  ightarrow\ \{\ \}\ )$ Returns the DEL $ ightarrow$ menu key.
27EAF	<skipkey< td=""><td><math>( \rightarrow \{ \} )</math> Returns the <math>\leftarrow</math>SKIP menu key.</td></skipkey<>	$( \rightarrow \{ \} )$ Returns the $\leftarrow$ SKIP menu key.
27EFB	>SkipKey	$( \rightarrow \{ \} )$ Returns the SKIP $\rightarrow$ menu key.
2EEE6	InitEd&Modes	$( \rightarrow )$ :: InitEdLine InitEdModes;
2EEE7	InitEdLine	$( \rightarrow )$ :: DEL_CMD ;
2EEE8	InitEdModes	$( \rightarrow )$
2F05E	SaveLastEdit	( $\$ $\rightarrow$ ) Calls CMD_STO if history is on.
2F326	CMDSTO	( $\$ \to$ ) Adds string to the list of the last 4 commands, accessible with the CMD key.

# **Chapter 39 Plotting**

The commands in this chapter deal with aspects related to plotting. Entries here deal primarily with the PPAR variable, that contains the parameters used in plotting. This variable is a list with the following parameters:

 $\{(x_{\min}, y_{\min}) (x_{\max}, y_{\max}) \text{ indep res axes type depend}\}$ 

This is the meaning of each of the parameters:

Parameter	Description	Default value
$(x_{\min}, y_{\min})$	A complex number representing the coordi-	(-6.5, -3.1)
	nates of the lower left viewing range.	
$(x_{\max}, y_{\max})$	A complex number representing the coordi-	(6.5, 3.2)
	nates of the upper right viewing range.	
indep	The independent variable.	X
res	Resolution. A number that represents the	0
	interval between the plotted points.	
axes	A complex number that represents the co-	(0,0)
	ordinates of the intersection of the axes. It	
	can also be a list representing this coordi-	
	nate and many other details, which are not	
	described in this book.	
type	The name of the (user) command that speci-	FUNCTION
	fies the plot type.	
depend	Dependent variable.	Y

316 39. Plotting

# 39.1 Reference

Addr.	Name	Description
2F162	CHECKPICT	( $\rightarrow$ ) Checks size of GBUFF. If it is smaller than 131x64 sets GBUFF back to its default size (131x64).
2EF06	CKPICT	( $xPICT \rightarrow$ ) Checks for user word $xPICT$ on level 1. Errors (SETTYPEERR) if there is another object.
2F258	PICTRCL	( xPICT $\rightarrow$ grob ) Does CKPICT, then recalls GBUFF and does TOTEMPOB.
2F355	MAKEPVARS	( $\rightarrow$ { } ) Creates the default PPAR variable in the current directory and returns its value.
2F163	CHECKPVARS	( → {})  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.
2F33D	GETPARAM	( $\# \to \text{ob}$ ) Extracts the $\#$ th item from PPAR. No error checking!
2F0FF	GETXMIN	( $\rightarrow$ % ) Recalls XMIN from the PPAR list if existent. If not, the default PPAR is created in the current directory.
2F366	PUTXMIN	( $\% \rightarrow$ ) Sets a new value for XMIN. PPAR is created if necessary.
2F0FE	GETXMAX	( $\rightarrow$ % ) Recalls XMAX from the PPAR list if existent. If not, the default PPAR is created in the current directory.
2F365	PUTXMAX	( $\$ \rightarrow$ ) Sets a new value for XMAX. PPAR is created if necessary.

Addr.	Name	Description
2F100	GETYMIN	( → % )
		Recalls YMIN from the PPAR list if existent. If
		not, the default PPAR is created in the current
		directory.
2F368	PUTYMIN	$( \ \ \ \ \ \ \ \ \ \ \ )$
		Sets a new value for YMIN. PPAR is created if
		necessary.
2F10E	GETYMAX	$( \rightarrow \% )$
		Recalls YMAX from the PPAR list if existent. If
		not, the default PPAR is created in the current
		directory.
2F367	PUTYMAX	( % → )
		Sets a new value for YMAX. PPAR is created if
		necessary.
2F107	GETPMIN&MAX	( → C% C% )
		Returns PMIN and PMAX.
2EEF2	PUTINDEP	$( ID \rightarrow )$
		Internal $xINDEP$ if the arg is an ID.
2EEF3	PUTINDEPLIST	$(\ \{\}\  o\ )$
		Internal $xINDEP$ if the arg is a list.
2F0E8	INDEPVAR	$( \rightarrow id )$
		Recalls the independent variable. If a list, ex-
		tract first element.
		:: GETINDEP DUPTYPELIST? ?CARCOMP ;
2F106	GETINDEP	$(\rightarrow id)$
		$( \rightarrow \{\})$
		Recalls the independent variable field in PPAR.
2EEF5	GETPTYPE	( $ ightarrow$ name )
		Recalls the plot type using GETPARAM.
2EEF6	PUTPTYPE	( name $\rightarrow$ )
		Sets a new plot type. PPAR is created if neces-
		sary.
2F10D	GETRES	$( \rightarrow \% )$
		Recalls the plot resolution using GETPARAM.
2EEF4	PUTRES	$(\% \rightarrow)$
		Set new plot resolution. PPAR is created if nec-
		essary.
2F33E	GETSCALE	$( \rightarrow \% \%' )$
		Recalls the plot scale parameters.

318 39. Plotting

Addr.	Name	Description
2EEF1	PUTSCALE	( % %' → )
		Set new plot scale. PPAR is created if necessary.
2EEEF	AUTOSCALE	$( \rightarrow )$
		Internal AUTO.
2EF60	DOGRAPHIC	$( \rightarrow )$
		Sets the scroll mode of PICTURE and is essen-
		tially the same as $\{\ \}$ PVIEW.
25ECF	EQUATION	$( \rightarrow ob )$
		Recall the current equation, stored in the 'EQ'
		variable.
2F339	${ t GetEqN}$	$( \#n \rightarrow ob T )$
		( $\#n \rightarrow NULL\$ F$ )
		Get the #nth equation, if EQ is a list of equations.
25EB5	DORCLE	$( \rightarrow ob )$
		Recalls the contents of the EQ variable, errors if
		it does not exist.
25EB6	DOSTOE	$( ob \rightarrow )$
		Stores ob into the variable EQ.
2F297	XEQPURGEPICT	$(xPICT \rightarrow )$
		If object in level one is xPICT, erases the graphic
		display. Otherwise, errors.
2F105	GDISPCENTER	$(\rightarrow)$
		Moves to center of graphics display
2EF01	DOPX>C	$( \{ \text{ hxs hxs'} \} \rightarrow \text{C%} )$
		Converts a list of two hex strings into a complex
		number. Used for plotting coordinates. Inverse
		operation is DOC>PX.
2EF02	DOC>PX	$( \ \texttt{C\$} \ \rightarrow \ \{ \ \texttt{hxs} \ \texttt{hxs'} \ \} \ )$
		Converts a complex coordinate point into list
		of two HXS numbers. Inverse operation is
		DOPX>C.

# Part IV

# The HP49 CAS

# Chapter 40 Introduction to the HP49 CAS

One of the major innovations in the HP49G is the powerful built-in Computer Algebra System (CAS). The HP49G CAS is derived mainly from the ALG48 and ERABLE libraries, originally written for the HP48 calculators. But on the HP49G, the CAS is fully integrated into the operating system, so that User RPL operators transparently access the CAS if the arguments require it. A huge number of supported entry points give access to the internal commands of the CAS, enabling users to write their own programs and commands dealing with symbolic objects, matrices and infinite precision integers.

#### 40.1 Problems with These Chapters

The initial version of the reference listing of CAS commands for this book was derived from the source files¹ of ALG48 and ERABLE. The problem with this approach is that in the source, the different entries are not fully ordered according to functionality. Rather, each source file handles a certain area of CAS commands, and many utility routines are included inside the same file. For this reason there are several different locations where for example meta-object handling routines may be found. There are even similar such routines (which seem to do the same thing) in different files. We have made a significant effort to reorder the entries by functionality, but we realize that we have only partially succeeded. A deeper knowledge of the CAS and its internals is needed to complete this work.

A full documentation of the CAS should also contain extensive material about the internal representation of CAS objects, and many examples how to use these commands. Let us hope that Bernard Parisse will one day find time to fully document the HP49G CAS internals. For the time being we include a slightly edited version of a document he provided to us, which introduces some important aspects of the CAS. The reminder of this part will then just be a reference list of entries.

 $<sup>^{1}</sup>$ actually, from a condensed version of the routine headers provided to us by Bernard Parisse

#### 40.2 Symbolic Objects

The CAS manipulates symbolic scalars and vectors or matrices of these objects. Symbolic scalars have 3 representations, which we show in the following table using '2X' as an example expression.

user representation a SYMBOL object which is the composite object. For

the example expression it looks like this:

SYMBOL Z2 ID X x\* ;

meta representation the SYMBOL object exploded onto the stack. For

'2\*X', these are the 4 objects Z2 ID  $X \times * #4$  on stack

levels 4 to 1.

list representation polynomial coefficients (in the example: { 2 0 })

with respect to the list of variables ( $\{X\}$ ).

Conversion from user to meta representation is done by SYMBINCOMP (a generalized INNERCOMP to handle non symbolic objects like integers).

Meta representation is used to handle operations when rational normal form is not relevant. It is more efficient than symbolic representation because you do not have to explode and rebuild the symbolic objects, everything is done on the stack. Stack operations on metas are described in Chapter 12. Unary and binary operators are often the operator name prefixed by addt (e.g. addtSIN). An example for a complex routine working on meta objects is CASCOMPEVAL. It does a COMPEVAL-like loop but with metas on the stack instead of symbolics.

The list representation is used when the rational normal form is important. This is the case for integration of rational fractions, rational simplifications, Laplace transformations, series expansions and similar operations. The first step for the conversion is to find the *list of variables* with respect to which the expression is rational. For example,

$$\frac{\sin(x) + y}{\cos(x) + y} \tag{40.1}$$

is rational with respect to  $\{\sin(x)\cos(x)y\}$ . Given a symbolic or a list/array of symbolic objects, the user word LVAR, or the System RPL command LVARext, returns this list of variables. The conversion is then done as a quotient of 2 multivariate polynomials with respect to this list of variables, with this ordering.

Gaussian integers are represented as secondaries with two elements:

:: imaginary\_part real\_part :. The imaginary and real parts must be integers.

Square roots are represented as irrquad: :: x« a b c x» ; represents a+b\*sqrt(c).

*Polynomials* are defined as a list of coefficients that are polynomials themselves, constants (integer or Gaussian integer) or irrquads. *Rational fractions* built over these polynomials are represented as SYMBOL num deno x/i; where num and deno are polynomials that are prime together (in exact mode).

The main conversion routine to the list format is VXXLext. The main back conversion routine is R2SYM. There are several specialized routines to convert a list or meta of symbolic objects, or to convert a symbolic object into meta-representation, or from list format to the meta-representation of a symbolic object. These specialized routines are more efficient but more difficult to use.

Rational operators on list objects are implemented (QAdd, QSub, QDiv, QMul, QNeg, RPext), as well as Euclidean divisions with specializations e.g. for integers or Gaussian integers.

#### 40.3 A Few Examples

In the following examples, the comments in each line represent the objects on the stack after the current command.

Rational simplification of a symbolic object might be coded as

```
1 :: (symb)
     FPTR2 ^LVARext (symb lvar)
     FPTR2 ^VXXLext (lvar n/d)
     FPTR2 ^R2SYM (symb)
5 ;
```

The scalar product of 2 symbolic vectors in "list form"

```
FPTR2 ^QMul

FPTR2 ^QAdd

10 LOOP

(y1, ..., yn-1, #n, X.Y)

OVER #1+UNROLL #1- NDROP

;
```

# **Chapter 41 Type Checking and Conversion**

The entries in this chapter are used to check for the special CAS objects described in Chapter 40, and to convert between this different kinds of objects.

### 41.1 Reference

Addr.	Name	Description
157006	^SYMBINCOMP	( symb $ ightarrow$ ob1 obN #n )
		( ob $\rightarrow$ ob #1 )
		( $\{\} \rightarrow \{\} \#1$ )
		Explodes symbolic object into meta. Other ob-
		jects are converted into one-object metas by
		pushing #1 into the stack.
12A006	^2SYMBINCOMP	( ob1 ob2 $ ightarrow$ meta1 meta2 )
		Does ^SYMBINCOMP for 2 objects.
4D7006	^VXXLext	( ob Lvar $ ightarrow$ Q )
		Converts object to internal form. The object
		can be a symbolic, a symbolic vector or a sym-
		bolic matrix. If the conversion was not suc-
		cessfull, vxxxlflag is cleared.
400006	^R2SYM	( lvar ob $\rightarrow$ ob )
		Back conversion of a scalar object.
4D8006	^METALISTVXXL	( Meta $ ightarrow$ Meta )
		Conversion of all elements of a meta object
		with respect to the variables in LAM1.
4D9006	^VXXLFext	$(n/d \rightarrow Z1/Z2)$
		Conversion of a fraction which does not de-
		pend on any variables.

Addr.	Name	Description
4DA006	^VXXL1ext	$(n \rightarrow Z)$
		Conversion of an object which does not de-
		pend on any variables.
4DB006	^VXXL0	( ob $\rightarrow$ Q )
		Conversion of object with respect to Lvar in
		LAM1.
4DC006	^VXXL2NR	( Meta $ ightarrow$ Q )
		Converts symbolic meta to internal form
		(LAM1=Lvar). Set nocareflag to avoid square
		root problems.
4DD006	^VXXL2	( Meta $ ightarrow$ Q )
		Converts symbolic meta to internal form
		(LAM1=Lvar).
167006	^TYPEIRRQ?	( ob $ ightarrow$ flag )
		Is ob an irrquad?
168006	^DTYPEIRRQ?	( ob $ ightarrow$ ob flag )
		DUP, then ^TYPEIRRQ?.
177006	^CKMATRIXELEM	( ob $ ightarrow$ ob )
		Checks that ob is a valid internal matrix ele-
		ment. Look for CK[]NCK for user matrix ele-
		ment.
18F006	^CKFPOLYext	( ob $ ightarrow$ ob )
		Errors if list contains secondaries or empty
		lists.
190006	^CK2FPOLY	( ob ob $ ightarrow$ ob ob )
		Does CKFPOLYext on two objects.
19E006	^CLEANIDLAM	$( ob \rightarrow ob )$
		Suppresses SYMB if not needed.

# **Chapter 42 Integers**

This chapter lists the functions that deal with Arbitrary Precision Integers, a new number type provided by the HP49 CAS. For a description of that type, see Chapter 5.

You will notice that there are no entries for basic arithmetic operations on integers. This is because there are no specific such entries for integers. Instead, use the polynomial entries like ^QAdd, ^QMul, etc. listed in Chapter 46.

#### 42.1 Reference

#### 42.1.1 Built-in Integers

Addr.	Name	Description
2E0006	^DROPZ0	$( ob \rightarrow z0 )$
2DF006	^DROPZ1	( ob $ ightarrow$ z1 )
392006	^2DROPZ0	$(21 \rightarrow z0)$
3B3006	^NDROPZ0	( obnob1 $\#n \rightarrow z0$ )
		Replaces meta with Z0.
3B4006	^NDROPZ1	( obnob1 $\#n \rightarrow z1$ )
		Replaces meta with Z1.

#### 42.1.2 Conversion Functions

Addr.	Name	Description
0EE006	^#>Z	( # -> Z )
		Converts bint to zint.
0F5006	^R>Z	$( % \rightarrow z )$
		Converts real to zint. Do not call this entry if
		the number if not an integer.

328 42. Integers

Addr.	Name	Description
18D006	^R2Zext	( % → %%/Z )
		Converts real to zint, or to long real if the
		number is not an integer. mode if number is
		not an integer.
0ED006	^H>Z	( HXS $ ightarrow$ Z / Error )
		Checks if HXS is a proper zint number and
		trims it.
0F2006	^S>Z	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
		Converts decimal in a string into a zint.
0F3006	^S>Z?	( $\$ \rightarrow z T$ )
		( $\$ \rightarrow \$ F$ )
		If possible, converts string into a zint and re-
		turns TRUE. If not, keeps the original string
		and returns FALSE.
184006	^CK1Z	( $\$/\#/hxs \rightarrow Z$ )
		Checks for an integer. Converts strings, bints
		or hxs's to zints. Errors for other object types.
185006	^CK2Z	( ob ob' $ ightarrow$ Z Z' )
		Like ^CK1Z, but for two objects.
186006	^CK3Z	( ob ob' ob'' $ ightarrow$ Z Z' Z'')
		Like ^CK1Z, but for three objects.
202006	^CK&CONVINT	( symb $ ightarrow$ zint )
		( symb $ ightarrow$ :: zint zint' ; )
		Check that a sym is a zint or Gauss integer,
		convert it.
203006	^CK&CONV2INT	( symb symb' $ ightarrow$ zint zint' )
		( symb symb' $ ightarrow$ :: zint1 zint2 ; ::
		zint3 zint4 ; )
		Check that 2 sym are zint or Gauss integer,
		convert them.
205006	^CONVBACKINT	( $zint   c \rightarrow symb$ )
204006	^CONVBACK2INT	( $zint c$ $zint c$ $ o$ $symb$ $symb$ )
0F4006	^Z>ZH	( $Z \rightarrow Z'$ )
		Converts decimal Z to hex Z.
18E006	^Z2Sext	$(Z \rightarrow '\$Z')$
		Converts Z to string number. The number is
		embedded in a symbolic to enable using it in
		algebraics.

# 42.1.3 General Integer Operations

Addr.	Name	Description
101006	^ZTrim	( $Z \to Z'$ ) Strips Z from unnecessary leading nibbles. Counts nibbles required for representation. If that equals
		used nibbles then quick exit. Else allocates new object, copies significant mantissa nibbles and apends original sign.
102006	^ZAbs	( $z \rightarrow  z $ ) Takes the absolute value of Z. If Z is already pos-
		itive then does nothing. Else duplicate object and
50B006	^ZABS	change sign. ( $Z \rightarrow Z'$ )
302000	21120	Absolute value.
0E0006	^ZSQRT	$(Z \rightarrow Z' \text{ flag })$
		Calculates integer part of square root. If the number was a square, then flag is TRUE to indicate that
		the returned result is exact.
3D0006	^Mod	$( \ Z \ Zn \  ightarrow \ Z' \ )$ Make ${f Z}$ modulo ${f N}.$
0DD006	^ZMod	wake Z modulo N. ( $Z1 Z2 \rightarrow Z'$ )
105006	^ZNMax	$($ Z1 Z2 $\rightarrow$ NormMax[Z1,Z2] $)$
		Returns the integer with the greatest absolute value. (Returns Z1 if $ Z1  \ge  Z2 $ ; returns Z2 if
		value. (Returns 21 ii $ Z1  \le  Z2 $ , returns 22 ii $ Z1  <  Z2 $ ).
106006	^ZNMin	( $Z1 Z2 \rightarrow NormMin[Z1,Z2]$ )
		Returns the integer with the smallest absolute value. (Returns Z1 if $ Z1  \le  Z2 $ ; returns Z2 if
		Z1  >  Z2 ).
10D006	^ZBits	$(Z \rightarrow Z \text{ #bits })$
10E006	^ZBit?	Calculates number of bits used in Z. ( $Z \text{ #bit } \rightarrow Z \text{ flag }$ )
101000	ZDIC.	Tests if a bit in Z is set. Count starts from zero, as
	•	opposed to ZBits.
2B7006	^ZGCDext	( $Z2 Z1 \rightarrow Z$ ) Integer GCD.
2B8006	^ZGcd	( $Z2 Z1 \rightarrow Z$ )
		This is the same entry as ZGCDext.

330 42. Integers

Addr.	Name	Description
3D6006	^IEGCDext	( $a b \rightarrow d u v$ )
		Bezout for integers. d=au+bv=gcd(a,b).
3D9006	^INEGCD	( $a b \rightarrow d u v$ )
07C007	^#FACT	$( \ \# \  o \ Z \ )$
		Calculates the factorial of an integer. Works fine
		for all numbers #0 - #FFFFF, although at some
		point you will get an out of memory error.
576006	^factzint	$(z \rightarrow z!)$
		Factorial for long integers.
215006	^PA2B2	$(z/\% \rightarrow a+bi)$
		Internal PA2B2.

# **42.1.4** Integer Factorization and Prime Numbers

Addr.	Name	Description
0C9006	^ZFactor	( $Zs \rightarrow Lf$ )
		Factors signed long integer.
0CA006	^NFactor	$(z \rightarrow \{\})$
		Factors positive long integer.
0CB006	^NFactorSpc	$(z \rightarrow \{\})$
		Semi-factors positive long integer. This is
		regular factorization with an extra 'hopeless?'
		test.
0CD006	^SFactor	( S $ ightarrow$ Lf )
		Factors short integer. Pollard Rho, with the
		assumption that trial division has been done
		already. Thus any factor less than 4012009
		is known to be a prime, for greater factors a
		primality test is used before calling the actual
		Pollard Rho. Pollard Rho does not find the
		factors in order of magnitude, thus the results
		will be sorted after full factorization has been
		achieved.

Addr.	Name	Description
0CE006	^SPollard	(S $\rightarrow$ S1 S2) Factors short integer into 2 parts using Pollard Rho algorithm. Trial division and primality tests should be done prior to calling this subroutine, otherwise an eternal loop is risked. The random number generator is modeled after the user level RAND command,
0CF006	^BFactor	although the starting value is different. ( $N \to Lf$ ) Factors long integer. Brent-Pollard, with the assumption that trial division has been done already. When a small factor is found SFactor is called to get full short factorization. Since the factorization can potentially take a very long time, an execution time test is used to abort factoring very long integers (limit is 60s for each composite). The factors are sorted at exit.
OD0006	^BrentPow	( Za Z1 Z2 Zn $\#k \to Z$ ) Modular*+^mod for Brent-Pollard factorization. Output is Z1*Z2+Za mod Zn repeated k times Note that k=0 and k=1 give the same result. Also Z1 $\neq$ Z2 makes no sense for k $\neq$ 0. All arguments are assumed to be positive. Za is assumed to be < 16. In some instances k can be a very high number, thus it might make sense to use Montgomery multiplication.
0D1006	^ZPrime?	<ul> <li>( Z → flag )</li> <li>Primality test for a positive integer. According to Pinch commercial software packages use only about 5-10 bases by default, maximum around 25. The latest versions usually implement a deterministic.</li> </ul>
0D2006	^ZIsPrime?	( $Z \rightarrow flag$ ) Probabilistic primality test for a positive integer.

332 42. Integers

Addr.	Name	Description
0D3006	^SIsPrime?	( S $\rightarrow$ flag )
		Tests if positive short Z is prime. M-R test
		fails for integers $\leq$ 3, so we just test them
		separately at the start. For convenience lets
		define 0 and 1 to be primes also.
0D4006	^BIsPrime?	( S $\rightarrow$ flag )
		Test if positive long Z is prime.
0D5006	^BRabin	( Z $\#$ base $\rightarrow$ Z flag )
		Performs Miller-Rabin test for long positive
		integer. Returns TRUE if base witnesses com-
		posite. Else returns FALSE.
0D6006	^ZTrialDiv2	$(Z \rightarrow Z' \#n)$
		Remove factors of 2 from integer. #n is the
		power of two extracted from the number. The
		sign is also handled correctly, even though it
		is never required in ALG48 (absolute Z).
0D7006	^ZTrialPrime?	( $Z \rightarrow flag$ )
		Trial division primality test for a positive in-
		teger. works for $Z \ge 3$ (return false for $Z=2$ ).
0D8006	^ZTrialDiv	( $Z \rightarrow Mf Z'$ )
		Trial division of a positive integer. If Z' is one
		then full factorization was achieved. The long
		trial division is not too slow, since division by
		short integer is quite fast. The quotient is
		also checked so that a final factor less than
		2000 <sup>2</sup> will also be automatically detected.
0C7006	^Prime+	$(Z \rightarrow Z')$
		Returns next prime ( $Z' > Z$ ).
0C8006	^Prime-	$(Z \rightarrow Z')$
		Returns previous prime ( $\mathrm{Z'} < \mathrm{Z}$ ).

# 42.1.5 Gaussian Integers

Addr.	Name	Description
114007	^TYPEGAUSSINT?	( ob $\rightarrow$ flag )
		Checks if ob is Gaussian integer.
115007	^DTYPEGAUSSINT?	( ob $ ightarrow$ ob flag )
		Checks if ob is Gaussian integer.
116007	^DUPTYPEGAUSSINT?	( ob $ ightarrow$ ob flag )
		Checks if ob is Gaussian integer.
187006	^CK1Cext	( ob $ ightarrow$ flag )
		Checks if object is integer or Gaussian inte-
		ger.
15D006	^CXRIext	( C $ ightarrow$ Zre Zim )
		Returns real and imaginary part of Gaus-
		sian integer.
2B5006	^CGCDext	( C2 C1 $\rightarrow$ C )
		GCD for Gauss integers.
4D5006	^CSQFFext	( C $ ightarrow$ { factor1 mult1 $\ldots$ factn
		multn } )
		Factorization of Gauss integers. This is not
		the complete factorization of C over Gauss
		integers since the GCD of the real part and
		imaginary part of c is factored only over R.
4D4006	^SECOSQFFext	( :: $x << a b c x>> \rightarrow \{ fact1 \}$
		<pre>mult1 factn multn } )</pre>
		Factorization of irrquads and Gauss inte-
		gers.
4D6006	^SUMSQRext	$(Z \rightarrow Z C)$
		Returns a Gauss integer C so that $ C ^2=Z$ .
		Z must be 2 or so that Z=1 mod 4. If $Z \neq 1$
		mod 4, "Z is not 1 mod 4" error. Z should be
		prime to ensure the existence of a solution.
518006	^CNORMext	$(C \rightarrow  C ^2)$
		Square modulus of a Gauss integer.

334 42. Integers

# 42.1.6 Integer Tests

Addr.	Name	Description
265C1	Z=	( Z Z' $ ightarrow$ flag )
265C6	Z<>	( $Z$ $Z'$ $ o$ flag )
265BC	Z<	( $Z$ $Z'$ $ o$ flag )
265D0	Z<=	( $Z$ $Z'$ $ o$ flag )
265B7	Z>	( $Z$ $Z'$ $ o$ flag )
265CB	Z>=	( $Z$ $Z'$ $ o$ flag )
0F8006	^QIsZero?	( $Q \rightarrow flag$ )
		Tests if Q is zero. Assumes list contains only
		lists or hexes!.
0F7006	^DupQIsZero?	( $Q \rightarrow Q$ flag )
		Duplicates Q and tests if Q is zero. Assumes
		list contains only lists or hexes!.
0FA006	^ZIsOne?	$(Z \rightarrow flag)$
		Tests if Z is Z1.
0F9006	^DupZIsOne?	( $Z \rightarrow Z$ flag )
		Duplicates Z, and returns TRUE if Z is 1.
109006	^DupZIsTwo?	( $Z \rightarrow Z$ flag )
		Returns TRUE if Z is 2.
0FC006	^ZIsNeg?	$(Z \rightarrow flag)$
		Tests if Z is negative.
0FB006	^DupZIsNeg?	( $Z \rightarrow Z$ flag )
		Tests if Z is negative.
10A006	^DupZIsEven?	( $Z \rightarrow Z$ flag )
		Tests if Z is even.
107006	^ZNLT?	( Z1 Z2 $ ightarrow$ flag )
		TRUE if $ Z1  <  Z2 $ .
19A006	^OBJINT?	$(z/\% \rightarrow z \text{ flag })$
		Tests if Obj is an integer.
19B006	^OBJPOSINT?	( $z/% \rightarrow z$ flag )
		Tests if Obj is a positive integer smaller than
		Zsmall.
19C006	^CKINT>0	(Obj → Obj flag)
		Tests if Obj is a strictly positive integer.
198006	^METAINT?	( Meta → Meta flag )
		Tests if Meta is an integer.

Addr.	Name	Description
199006	^METAPOSINT?	( Meta $ ightarrow$ Meta flag )
		Tests if Meta is a positive integer smaller than
		Zsmall.
0CC006	^DupTypeS?	( $Z \rightarrow Z$ flag )
		Tests if Z is short ( $\leq$ 64 bits).

# Chapter 43 Matrices

The CAS' Symbolic Matrices are a new object on the HP49 used to represent matrices. Unlike the old array object present since the HP48, these matrices can have symbolic expressions inside them. It is also possible to have objects of different types inside the array.

This kind of matrix is actually a composite object, and you can use the functions of Chapter 11 on them.

The following disassembly of the matrix  $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$  should make it clear how to create one using MASD, and why they are actually composites:

It should also be noted that most (if not all) the functions described below for dealing with symbolic matrices also work with lists of lists. The reason should be obvious: the structure of these matrices and a list of list is the same, only the prolog address changes.

Some entries dealing with the old HP48 arrays, described in Chapter 10.1, also work with symbolic matrices.

#### 43.1 Reference

## 43.1.1 Creating and Redimensioning Matrices

Addr.	Name	Description
371006	^MATIDN	( M/z/% → M' )
		Creates identity matrix.
372006	^MATCON	( M ob $\rightarrow$ [ob] )
		Creates constant matrix from matrix.
373006	^MAKEARRY	( $\{\#el\}\ $ symb $ ightarrow$ [] )
		( $\{ \text{\#rows \#cols} \} \text{ symb }  o \text{[[]]}$ )
		Creates constant matrix/array from ob type.
345006	^DIMRANM	$(\ \{\}\  o\ M'\ )$
		Creates symbolic random matrix from dimen-
		sions.
344006	^MATRANM	$(M \rightarrow M')$
		Changes all elements of matrix to elements
		generated randomly.
374006	^OBJDIMS2MAT	( ob $\{\} \rightarrow M$ )
		Creates constant matrix from dimension and
		ob.
375006	^LCPROG2M	( $\#$ n $\#$ m prg $\rightarrow$ M )
		Fills a matrix of specified size using a pro-
		gram. prg must take two arguments and re-
		turn one argument. On entry MAKE2DMATRIX
	<b>^</b>	provide the indexes as Z integers.
376006	^MAKE2DMATRIX	$( #n #m prg \rightarrow M )$
		Creates matrix from size and program (with
		stack checking). prg must take 2 args and re-
		turn 1 arg. On entry MAKE2DMATRIX provide
277006	^ 1 0 -1 +1	the indexes as Z integers.
377006	^make2dmatrix	( #n #m prg → meta-M )
		Create meta-matrix from size and program (with stack checking). prg must take 2 args
		and return 1 arg On entry make2dmatrix
		provide the indexes as Z integers.
341006	^MATREDIM	provide the indexes as Z integers.  ( $M \{\} \rightarrow M'$ )
241000	HATKEDTH	Changes size of a matrix, removing elements
		and/or adding zeros, as necessary.
		and or adding Leros, as necessary.

338 43. Matrices

Addr.	Name	Description
342006	^VRRDM	$( []/[[]] \{\} \rightarrow [] )$
		Vector Right ReDiMension: adds 0 to the right.
343006	^VRRDMmeta	( meta $\#1 \rightarrow meta-\#1$ ) Meta Right ReDiMension: adds 0 to the right.

#### 43.1.2 Conversion

Addr.	Name	Description
16A006	^{}TO[]	$(\ \{\}\ \rightarrow\ [\ ]\ )$
		Converts from list-of-lists representation to
		matrix. No checks on the element type.
17A006	^LIST2MATRIX	$(\ \{\}\  o\ [\ ]\ )$
		$(\{\{\}\} \rightarrow [[]])$
		$( ob \rightarrow ob )$
		Converts a symbolic list to a matrix. Does
		not check that matrix is a valid one. Use
167006	^ [ ] == ( )	DTYPFMAT? to do that.
T6B006	^[]TO{}	$( [] \rightarrow \{\} )$ Converts from matrix to list-of-lists.
170006	^MAMDIXOITOM	
1/9006	^MATRIX2LIST	$( [] \rightarrow \{ \} )$ $( [[]] \rightarrow \{ \{ \} \} )$
		$( op \rightarrow op )$
		Converts a symbolic matrix to a list.
17E006	^ARRAY2MATRIX	( [ ] → [ ] )
1,1000		$([[]] \rightarrow [[]])$
		Converts array to symbolic array if necessary.
175006	^SAMEMATRIX	( M1 M2 $\rightarrow$ M1 M2 flag )
		If one object is a symbolic array, converts both
		arrays to symbolic form. Returns TRUE for
		symbolic matrices and FALSE for numeric.
176006	^SAMEMATSCTYPE	( M ob $ ightarrow$ M ob flag )
		If M is a numeric matrix and ob is not float,
		converts matrix to symbolic form. Returns
		TRUE for symbolic and FALSE for numeric.
003007	^ArryToList	$([]/[[]] \rightarrow {}/{\{\}})$
		Converts normal array to list of lists; errors
		for symbolic arrays.

Addr.	Name	Description
17D006	^MATEXPLODE	( [[ob1obn]] → ob1obn
		[[ob1obn]] )

#### 43.1.3 Tests

Addr.	Name	Description
16C006	^DUPNULL[]?	( ob $ o$ ob flag )
		Tests for a null array.
359006	^NULLVECTOR?	( $V \rightarrow flag$ )
		Returns true if vector is null.
16F006	^CKSAMESIZE	( arryl arry2 $ ightarrow$ arry1 arry2 flag )
		Tests if arry1 and 2 have the same size.
170006	^DTYPENDO?	( ob $ ightarrow$ ob flag )
		Tests if object is a square symbolic matrix.
		Convert numeric array to symbolic matrix.
173006	^2DMATRIX?	( ob $ ightarrow$ ob flag )
		Tests if object is a 2D matrix.

#### 43.1.4 Calculations with Matrices

Addr.	Name	Description
320006	^MAT+	( M2 M1 $\rightarrow$ M2+M1 )
321006	^MADD	( M2 M1 $\rightarrow$ M2+M1 )
322006	^MAT-	( M2 M1 $\rightarrow$ M2-M1 )
323006	^MSUB	( M2 M1 $\rightarrow$ M2-M1 )
324006	^VADD	( V2 V1 $\rightarrow$ V2+V1 )
325006	^VSUB	( V2 V1 $\rightarrow$ V2-V1 )
326006	^MAT*	( M2 M1 $\rightarrow$ M2*M1 )
		Matrix product with size and type checking.
327006	^MMMULT	( M2 M1 $\rightarrow$ M2*M1 )
328006	^MVMULT	( M V $ ightarrow$ V' )
		Product of matrix by vector.
329006	^SCL*MAT	( ob M $ ightarrow$ M*ob )
		Scalar times matrix.
32A006	^MAT*SCL	( M ob $\rightarrow$ M*ob )
		Matrix times scalar.

340 43. Matrices

Addr.	Name	Description
32B006	^VPMULT	( $V  ext{ ob }  o V'$ )
		Multiplies vector by a scalar.
335006	^MATSQUARE	$(M \rightarrow M*M)$
32C006	^MAT^	( M $z/% \rightarrow M'$ )
		Integral matrix power.
32D006	^MATCROSS	$( \hspace{.15cm} [\hspace{.15cm}] $
		Vector product.
32E006	^MATDOT	( V2 V1 $\rightarrow$ ob )
		Scalar product with checking.
32F006	^RNDARRY	$(M \% \rightarrow M)$
		Rounds array.
330006	^TRCARRY	$(M \% \rightarrow M)$
		Truncates array.
332006	^MAT/SCL	( M ob $\rightarrow$ M/ob )
		Divides matrix by scalar.
333006	^MAT/	( $V M \rightarrow M^-1*V$ )
		"Divides" Vector by matrix.
334006	^MATCHS	( M $\rightarrow$ -M )
34E006	^MATINV	( $M \rightarrow M^-1$ )
336006	^MATCONJ	( $M \rightarrow M'$ )
337006	^MATRE	( $M \rightarrow re[M]$ )
338006	^MATIM	$(M \rightarrow im[M])$
339006	^MATTRACE	( M $ ightarrow$ trace )
		Matrix trace.
33A006	^MATTRN	( $M \rightarrow M'$ )
		Matrix transposition and conjugation.
33C006	^mattran	( M $ ightarrow$ Meta-M' )
		Transposes matrix, returns meta-matrix.
33D006	^mattrn	( Meta-M $ ightarrow$ Meta-M' )
		Transposes meta-matrix.
346006	^MATDET	( M $ ightarrow$ det )
		Determinant, expanding all (not row reduction).
347006	^MATRDET	( M $ ightarrow$ det )
		Determinant using row reduction.
348006	^MATFNORM	( $M \rightarrow ob$ )
		Frobenius norm.
349006	^MATRNORM	( $M \rightarrow ob$ )
		Row norm.

Addr.	Name	Description
34A006	^MATCNORM	$(M \rightarrow ob)$
		Column norm.
174006	^MATRIXDIM	( ob $\rightarrow$ $\#$ )
		Returns symbolic matrix dimensionality of an ob-
		ject.

## 43.1.5 Linear Algebra and Gaussian Reduction

Addr.	Name	Description
34C006	^MATREF	( $M \rightarrow M'$ )
		Returns matrix in Row-Echelon form.
34B006	^MATRREF	( $M \rightarrow M'$ )
		Returns matrix in Reduced Row-Echelon form.
34F006	^MATREFRREF	( M $\#$ full_ref $ o$ M list M' )
		If #full_ref is 1, returns Reduced Row-Echelon
		form, otherwise returns just Row-Echolong
		form.
367006	^MATRIXRCI	( ncol i M const $ ightarrow$ M' )
		Multiplies row #i of symbolic matrix M by con-
		stant. ncol is not used, it's here because of the
		stack state at call-time from inside laRCI.
368006	^MATRIXRCIJ	( ncol #i #j M const $ ightarrow$ M' )
		Does Lj <- c*Li+Lj. ncol is not used, it's here be-
		cause of the stack state at call-time from inside
		laRCI.
350006	^INXREDext	( Lvar $\#$ full_ref $M \to L$ var pivot $M$ )
351006	^METAMATRED	( Meta-M Lvar $\#full\_red \rightarrow meta-M$
		Lvar pivot )
352006	^METAPIVOT	( meta-M $\#1\ \#c \rightarrow$ meta-M $\#1\ \#1'\ \#c'$
		flag )
		Searchs a pivot in column #c starting from row
		#l. Flag is FALSE if pivot is not found. If pivot is
		found #l' is the row, #c is updated to #c'.
354006	^PIVOTFLOAT	( float $\rightarrow$ float_modulus )
34D006	^MATRANK	$(M \rightarrow Z/%)$
		Rank of a matrix.

342 43. Matrices

# 43.1.6 Linear System Solver

Addr.	Name	Description
080007	^LINSOLV	( b a → y )
		Solves $y'=ay+b$ .
0F4007	^SOLVEMETASYST	( meta-M $ ightarrow$ d meta-sol T )
		( meta-M $ ightarrow$ F )
		Solves linear system in meta representation.
		Meta-sol has been reduced to the same de-
		nominator d.
0F5007	^REDUCEMETASYST	( $meta-M \rightarrow meta->M'$ )
		Reduces linear system in meta representa-
		tion.
0F6007	^REDUCEMETAPSYST	( meta-M → meta-M' )
		Reduces linear system in meta representa-
		tion. Does not reduce last column of meta-
		matr. This is useful to solve linear system
	<b>^</b>	with parameters in the last column.
0F'/00'/	^SOLVECRAMER	( $meta-M \rightarrow d meta-sol T$ )
		( meta-M $\rightarrow$ F )
		Solves cramer system. Meta-matr must be
		fully reduced. Meta-sol is reduced to the
		same denominator. d flag is FALSE if dimen-
255006	^ G37 GM	sion do not match.
355006	^SYSText	( M linc → linc linc' res cas_p )
356006 357006	^STOSYSText ^MAKESYSText	$(M2 M1 \rightarrow M2 list)$
337000	MAVEDIBLEXC	( $M_{eq} M_{inc} \rightarrow M_{eq} M $ lidnt flag ) Converts linear equations to a matrix and
		checks that equation are linear with respect
		to lidnt.
		to man.

## 43.1.7 Other Matrix Operations

Ad	ldr.	Name	Description
35	A006	^FINDELN	( $\{\}$ A $ ightarrow$ # flag )
			Returns index # of element {} in array.

Addr.	Name	Description
35B006	^PULLEL[S]	( A $\#$ $\to$ A el )
		Extracts element of index # from array. Array
		type test is made in assembly for array speed.
35C006	^BANGARRY	( el $\#$ M $ ightarrow$ M' )
		Puts el at index # of matrix M.
35D006	^PUT[]	( el #i V $ ightarrow$ V )
		Replaces #i-th vector component by element.
17B006	^LENMATRIX	( [] $ ightarrow$ #el )
		$([[]] \rightarrow \#row)$
33E006	^MATSUB	( M rmin nrows cmin ncols { #m #n }
		$\rightarrow$ M')
		Extracts submatrix from a matrix.
340006	^MATREPL	( M1 M2 $\rightarrow$ M2' )
		Replaces part of matrix destination (M2) by
		matrix source (M1). LAM1 to 9 must be
		bound like in Llib/LIMain.s (9:r 8:c 7:dmat?
		6:f 5:md 4:nd 3:smat? 2:ms 1:ns ). Copy be-
	<b>.</b>	gins in matrix d at row r and column c.
35F006	^MATRIX>DIAG	( A ncols+1 ndiags $\rightarrow$ V )
		Extracts diagonal terms. ncols+1 is there
		because MATRIX>DIAG is called inside
260006	^143 == = = = = = = = = = = = = = = = = =	la>DIAG.
360006	^MATRIXDIAG>	( ncol+1 diagV dlen dims{} → M )
		Constructs a matrix from a vector of diagonal
261006	^]	terms.
361006	^la+ELEMsym	( V ob %i → V' )
262006	^INSERTROW[]	Inserts element in symbolic vector at row %i. ( $V \circ b \# i \rightarrow V$ )
362006	INSEKIKOW[]	$(V \cup D + I \rightarrow V)$ $(M \cup V + I \rightarrow M')$
		Inserts element/vector in symbolic vec-
		tor/matrix at row #i. Checks for $0 < \#i < \#n$
		+ 1, but does not check for matrix/vector size.
363006	^insertrow[]	( ob #i meta → meta )
303000	TILDOT CT OW[]	Inserts element/vector in meta-object at posi-
		tion #i. Checks for $0 < \text{#i} < \text{#n} + 1$ , but does
		not check for vector size.

344 43. Matrices

Addr.	Name	Description
364006	^INSERTCOL[]	( M V #i → M' )
		Inserts vector in symbolic matrix at col #i. Checks for $0 < \#i < \#n + 1$ , but does not check
		for matrix/vector size.
365006	^INSERT[]ROW[]	( M3 M2 $\sharp$ i $\to$ M )
303000	INSERT[]ROW[]	Inserts matrix2 in matrix3 starting from row
		#i. Checks for $0 < \#i < \#n+1$ , but does not
		check for matrix size.
366006	^INSERT[]COL[]	( M3 M2 $\#$ i $\rightarrow$ M )
		Inserts matrix2 in matrix3 starting from row
		#i. Checks for $0 < \#i < \#n + 1$ , but does not
		check for matrix size.
369006	^MATRIXCSWAP	( M #c #c' $\rightarrow$ M )
		Exchanges columns c and c' of a symbolic matrix.
36A006	^MATRIXRSWAP	( M $\#$ r $\#$ r' $\rightarrow$ M )
JOHOOO	MATICIZICOWAE	Exchanges lines r and r' of a symbolic matrix.
0AC003	^SWAPROWS	( M % %' $\rightarrow$ M' )
		SWAP two rows in matrix. Internal version of
		xRSWP.
36B006	^MATRIX-ROW	( M $\#r \rightarrow M'$ lr )
	_	Extracts row #r from M. Checks boundaries.
36C006	^METAMAT-ROW	( meta-M #r → meta-M lr )
		Extracts row #r from meta-matrix. Checks boundaries.
36D006	^MATRIX-COL	( M $\#$ c $\rightarrow$ M cc )
302000	MITTELL COL	Extracts column #r from matrix. Checks
		boundaries.
36E006	^METAMATCSWAP	( meta-M #c #c' $ ightarrow$ meta-M )
		Exchanges columns c and c' of a meta-matrix.
36F006	^METAMATRSWAP	( meta-M $\#1 \ \#1' \rightarrow meta-M$ )
		Exchanges lines l and l' of a meta-matrix (or
270006	^ CTOM	vector).
370006	^STOMAText	( $ exttt{M}  o  exttt{)}$ Stores matrix in 'MATRIX' in current direc-
		tory.
		••• J.

Addr.	Name	Description
378006	^ADDMATOBJext	( arry ob $\rightarrow$ arry arry )
		( ob arry $ ightarrow$ arry arry )
		Used for addition of numeric matrix and sym-
		bolic object.
379006	^VUNARYOP	( $v op \rightarrow V$ )
		Applies unary op(v[i]) to get V[i].
37A006	^VBINARYOP	( V2 V1 binop $\rightarrow$ V )
		Works even if V2 and V1 do not have not the
		same dimension.
37B006	^PEVAL	$(Vr \rightarrow P[r])$
		Horner evaluation, where elements of V rep-
		resent coefficients of a polynomial.

# 43.1.8 Eigenvalues, Eigenfunctions, Reduction

Addr.	Name	Description
37C006	^MATEGVL	( $M \rightarrow V$ )
		Computes eigenvalues of a matrix like EGVL.
37F006	^MATEGV	$(M \rightarrow V)$
		Computes eigenvalues/eigenvectors of a matrix
		like EGV.
37E006	^MADJ	( M $\rightarrow$ M^-1 P[M] P[lambda] )
		Computes inverse, matrix polynomial and char-
		acteristic polynomial.
380006	^JORDAN	( M $ ightarrow$ pmin pcar $\{ ext{evect}\}$ $\{ ext{eval}\}$ )
		( pmadj pcar $ ightarrow$ pmin pcar $\{ ext{evect}\}$
		{eval} )
		Eigenvalue/eigenfunctions computation.
22D006	^FLAGJORDAN	$(M \rightarrow )$
		Internal JORDAN.
381006	^QXA	( symb lidnt $ ightarrow$ M lidnt )
		Converts symbolic quad form to matrix quad
		form.
224006	^FLAGQXA	( symb lidnt $ ightarrow$ M lidnt )
		Internal QXA.
382006	^AXQ	( M lidnt $ ightarrow$ symb lidnt )
		Converts matrix quad form to qymbolic quad
		form.

346 43. Matrices

Addr.	Name	Description
225006	^FLAGAXQ	( M lidnt $ ightarrow$ symb lidnt )
		Internal AXQ.
383006	^GAUSS	( $symb \rightarrow D P symb'$ )
		Gauss reduction of quadratic form (symbolic).
226006	^FLAGGAUSS	( symb lidnt $ ightarrow$ symb' )
		Internal GAUSS.
384006	^SYLVESTER	( M $ ightarrow$ D P )
		Gauss reduction of a quadratic form (matrix).
227006	^FLAGSYLVESTER	( M $\rightarrow$ P D )
		Internal SYLVESTER.
228006	^PCAR	( [[]] $ ightarrow$ symb )
		Internal PCAR.

# **Chapter 44 Expression Manipulation**

The entries in this chapter are used for manipulation of expressions, when they are represented in their symbolic objects form. (See Chapter 45 for entries that deal with symbolics in Metaobject form). There are entries related to collection and expasion, trigonometric and exponential transformations and substitution of values in expressions.

#### 44.1 Reference

#### 44.1.1 Basic Operations and Function Application

Addr.	Name	Description
125006	^x+ext	( ob2 ob1 $\rightarrow$ ob2+ob1 )
		Symbolic addition, tests for infinities.
126006	^x-ext	( ob2 ob1 $\rightarrow$ ob2-ob1 )
		Symbolic subtraction, tests for infinities.
127006	^x*ext	( ob2 ob1 $\rightarrow$ ob2*ob1 )
		Symbolic multiplication, tests for infinities.
129006	^x/ext	( ob2 ob1 $\rightarrow$ ob2/ob1 )
		Symbolic division, tests for infinities.
12B006	^x^ext	( ob power $\rightarrow$ ob^power )
		Power.
12C006	^EXPAND^	$(x y \rightarrow x^y=exp[y*ln[x]])$
		Power with simplifications. If y is a fraction of
		integers, use XROOT instead.
4FB006	^QNeg	$(ob \rightarrow -ob)$
		Symbolic negation.
4FC006	^RNEGext	$(ob \rightarrow -ob)$
		Symbolic negation.

Addr.	Name	Description
4FA006	^SWAPRNEG	( ob2 ob1 $\rightarrow$ ob1 -ob2 )
		Does SWAP then symbolic negation.
4FE006	^RREext	( ob $\rightarrow$ Re(ob) )
		Symboloc real part.
4FD006	^SWAPRRE	( ob2 ob1 $\rightarrow$ ob1 Re(ob2) )
		SWAP, then RREext.
500006	^RIMext	$(ob \rightarrow Im(ob))$
		Symbolic imaginary part.
4FF006	^SWAPRIM	( ob1 ob2 $\rightarrow$ ob2 Im(ob1) )
		SWAP, then RIMext.
501006	^xREext	$(symb \rightarrow symb')$
		Complex real part. Expands only + - * / ^.
503006	^xIMext	( symb → symb' )
		Complex imaginary part. Expands only + - * / ^.
505006	^RCONJext	$( ob \rightarrow Conj(ob) )$
		Symbolic complex conjugate.
50D006	^xABSext	$(ob \rightarrow abs(ob))$
		Symbolic ABS function.
50A006	^RABSext	$(ob \rightarrow abs(ob))$
		Internal ABS. Internal representation.
52A006	^xINVext	( ob $\rightarrow$ 1/ob )
		Symbolic inversion.
557006	^xSYMINV	$(symb \rightarrow 1/symb)$
		Symbolic inversion.
553006	^xSQext	(symb → sq(symb))
		Symbolic square.
555006	^xSYMSQ	$(symb \rightarrow symb^2)$
51B006	^SXSQRext	$(ob \rightarrow sqrt(ob))$
		Does not take care of the sign.
51C006	^XSQRext	( ob $\rightarrow$ sqrt(ob) )
		Tries to return a positive square root if nocareflag
		is cleared.
52B006	^xvext	( ob $\rightarrow$ sqrt(ob) )
		Symbolic square root, tests for 0 and 1.
552006	^xSYMSQRT	$(symb \rightarrow sqrt(symb))$
521006	^CKLN	$(ob \rightarrow ln(ob))$
		Symbolic LN with special handling for fractions.
		Does not use the internal representation.

Addr.	Name	Description
522006	^xLNext	( ob $\rightarrow$ ln(ob) )
		Symbolic LN, without fraction handling.
525006	^EXPANDLN	$(ob \rightarrow ln(ob))$
		Symbolic LN using internal representation. Be-
		fore switching to internal representation, test for
		ABS, 0 and 1 and, in real mode, test if $ob=exp(x)$ .
528006	^REALLN	$(ob \rightarrow ln(ob))$
		Internal natural logarithm for a real argument.
526006	^CMPLXLN	$(ob \rightarrow ln(ob))$
		Internal complex natural logarithm.
527006	^LNATANext	$(ob \rightarrow ln(ob))$
		Internal natural logarithm for complex.
529006	^xEXPext	$(ydn \rightarrow exp(y*n/d*i*\pi))$
		Symbolic EXP, tests for 0, infinity and $i*k*\pi/12$
		where k is an integer. Tests for $d=1,2,3,4,6$ .
52C006	^xCOSext	$(ob \rightarrow cos(ob))$
		Symbolic COS, tests for 0 and multiples of $\pi/12$ .
		Also tests if $ob=acos(x)$ or $ob=asin(x)$ .
536006	^xSYMCOS	$(ob \rightarrow cos(ob))$
533006	^xACOSext	$( ob \rightarrow acos(ob) )$
		Symbolic ACOS. Tests for 0, infinity and tables.
53F006	^xSYMACOS	$(ob \rightarrow acos(ob))$
52D006	^xSINext	$(ob \rightarrow sin(ob))$
		Symbolic SIN, tests for 0 and multiplies of $\pi/12$ .
		Also tests if $ob=acos(x)$ or $ob=asin(x)$ .
538006	^xSYMSIN	$( ob \rightarrow sin(ob) )$
532006	^xASINext	$(ob \rightarrow asin(ob))$
		Symbolic ASIN. Tests for 0, infinity and tables.
53D006	^xSYMASIN	$(ob \rightarrow asin(ob))$
52E006	^xTANext	$( ob \rightarrow tan(ob) )$
		Symbolic TAN. Tests for 0 and multiplies of $\pi/12$ .
	_	Also tests if ob=atan(x).
53A006	^xSYMTAN	$(ob \rightarrow tan(ob))$
534006	^xATANext	$( ob \rightarrow atan(ob) )$
E 44 00 5	<b>^ ~~~</b>	Symbolic ATAN. Tests for 0, infinity and tables.
541006	^xSYMATAN	$(ob \rightarrow atan(ob))$
52F006	^xCOSHext	$(ob \rightarrow cosh(ob))$
E 4 E 2 2 6	A grn.=====	Symbolic COSH. Tests for 0, infinity and acosh(x).
545006	^xSYMCOSH	$(ob \rightarrow cosh(ob))$

Addr.	Name	Description
54E006	^xACOSHext	( symb $\rightarrow$ acosh(symb) )
		Symbolic ACOSH.
550006	^xSYMACOSH	$( \text{symb} \rightarrow \text{acosh}(\text{symb}) )$
530006	^xSINHext	$( ob \rightarrow sinh(ob) )$
		Symbolic SINH. Tests for 0, infinity and asinh(x).
543006	^xSYMSINH	$( ob \rightarrow sinh(ob) )$
54B006	^xASINHext	$(symb \rightarrow symb')$
		Symbolic ASINH.
54D006	^xSYMASINH	$( symb \rightarrow asinh(symb) )$
531006	^xTANHext	$(ob \rightarrow tanh(ob))$
		Symbolic TANH. Tests for 0 and atanh(x).
547006	^xSYMTANH	$(ob \rightarrow tanh(ob))$
		Symbolic TANH.
548006	^xATANHext	$(symb \rightarrow symb')$
		Symbolic ATANH.
54A006	^xSYMATANH	$(ob \rightarrow atanh(ob))$
55F006	^xSYMFLOOR	$(symb \rightarrow symb')$
561006	^xSYMCEIL	( $symb \rightarrow symb'$ )
563006	^xSYMIP	( $symb \rightarrow symb'$ )
565006	^xSYMFP	( $symb \rightarrow symb'$ )
567006	^xSYMXPON	( $symb \rightarrow symb'$ )
569006	^xSYMMANT	$(symb \rightarrow symb')$
56B006	^xSYMLNP1	$(symb \rightarrow symb')$
56D006	^xSYMLOG	$(symb \rightarrow symb')$
56F006	^xSYMALOG	$(symb \rightarrow symb')$
571006	^xSYMEXPM1	( $symb \rightarrow symb'$ )
572006	^factorial	$(symb \rightarrow symb!)$
		Symbolic factorial.
573006	^facts	$(\text{symb} \rightarrow \text{symb!})$
		Symbolic factorial.
575006	^xSYMFACT	$(\text{symb} \rightarrow \text{symb!})$
578006	^xSYMNOT	$(symb \rightarrow symb')$
128006	^x=ext	( ob2 ob1 $\rightarrow$ ob2=ob1 )

#### 44.1.2 Trigonometric and Exponential Operators

Addr.	Name	Description
408006	^COS2TAN/2	( $symb \rightarrow symb'$ )
		$x \rightarrow (1-(\tan(x/2))^2)/(1+(\tan(x/2))^2)$
40B006	^SIN2TAN/2	( $symb \rightarrow symb'$ )
		$x \rightarrow 2 \tan(x/2)/(1+(\tan(x/2))^2)$
40E006	^TAN2TAN/2	( $symb \rightarrow symb'$ )
		$x \rightarrow 2 \tan(x/2)/(1-(\tan(x/2))^2)$
412006	^COS2TAN	( $symb \rightarrow symb2$ )
		$x \rightarrow 1/sqrt(1+(tan(x))^2)$
414006	^SIN2TAN	( $symb \rightarrow symb'$ )
		$x \rightarrow tan(x)/sqrt(1+(tan(x))^2)$
41A006	^LNP12LN	$(symb \rightarrow symb')$
		$x \rightarrow ln(x+1)$
41B006	^LOG2LN	$(symb \rightarrow symb')$
		$\mathbf{x}  ightarrow \log(\mathbf{x})$
41C006	^ALOG2EXP	( symb → symb' )
		$x \rightarrow alog(x)$
41D006	^EXPM2EXP	$(symb \rightarrow symb')$
		$x \rightarrow \exp(x)-1$
41E006	^SQRT2LNEXP	$(symb \rightarrow symb')$
		$x \to \exp(\ln(x)/2)$
41F006	^sqrt2lnexp	$(\text{meta} \rightarrow \text{meta'})$
40000	^	$x \to \exp(\ln(x)/2)$
420006	^TAN2EXP	$(symb \rightarrow symb')$
400006	^ 7 G T 1 O T 1 T	$\mathbf{x} \rightarrow (\exp(\mathrm{i}2\mathrm{x})\text{-}1)/(\mathrm{i}^*(\exp(\mathrm{i}2\mathrm{x})\text{+}1))$
422006	^ASIN2LN	$(symb \rightarrow symb')$ $\vdots *!a(r \cdot sort(r^2, 1)) \cdot ri/2$
424006	^ 7 (10 (10 ) 1	$x \rightarrow = i*ln(x+sqrt(x^2-1))+pi/2.$ ( symb $\rightarrow$ symb')
424006	^ACOS2LN	$x \rightarrow \ln(x+\operatorname{sqrt}(x^2-1))/i$
427006	^TAN2SC	$\begin{array}{c} x \to \Pi(x+sqru(x 2-1))/I \\ \text{(symb } \to \text{symb')} \end{array}$
42/000	TANZSC	$(\text{Symb} \rightarrow \text{Symb})$ $x \rightarrow \sin(x)/\cos(x)$
42A006	^SIN2TC	$( symb \rightarrow symb' )$
12A000	DINZIC	$x \rightarrow \cos(x)*\tan(x)$
42C006	^COS2ext	$(symb \rightarrow symb')$
12000	CODZCAC	$x \rightarrow \operatorname{sqrt}(1-(\sin(x))^2).$
42E006	^SIN2ext	$(symb \rightarrow symb')$
121000	DINZCAC	$x \rightarrow \operatorname{sqrt}(1-(\cos(x))^2).$
		~9· ·· · · · · · · · · · · · · · · · ·

Addr.	Name	Description
431006	^ATAN2ASIN	( $symb \rightarrow symb'$ )
		$x \rightarrow asin(x/sqrt(x^2+1))$
434006	^ASIN2ATAN	( $symb \rightarrow symb'$ )
		$x \rightarrow atan(x/sqrt(1-x^2))$
437006	^ASIN2ACOS	( $symb \rightarrow symb'$ )
		$\mathbf{x} \to \pi/2\text{-}\mathrm{acos}(\mathbf{x})$
43C006	^ACOS2ASIN	$(symb \rightarrow symb')$
		$x \to \pi/2$ -asin(x)
43D006	^ATAN2LNext	( $symb \rightarrow symb'$ )
		$x \rightarrow i/2*ln((i+x)/(i-x))$
440006	^TAN2SC2	$(symb \rightarrow symb')$
		$x \rightarrow (1-\cos(2x))/\sin(2x)$
442006	^TAN2CS2	$(symb \rightarrow symb')$
		$x \rightarrow \sin(2x)/(1+\cos(2x))$
444006	^SIN2EXPext	$(symb \rightarrow symb')$
		$x \rightarrow (e^{(i*x)-1/e^{(i*x))/(2i)}$
446006	^COS2EXPext	$(symb \rightarrow symb')$
		$x \rightarrow (e^{}(i^*x)+1/e^{}(i^*x))/2$
448006	^SINH2EXPext	$(symb \rightarrow symb')$
		$x \rightarrow (e^x-1/e^x)/2$
44A006	^COSH2EXPext	$(symb \rightarrow symb')$
		$x \rightarrow (e^x+1/e^x)/2$
44C006	^TANH2EXPext	( $symb \rightarrow symb'$ )
		$x \rightarrow (e^2x-1)/(e^2x+1)$
44E006	^ASINH2LNext	( $symb \rightarrow symb'$ )
		$x \rightarrow ln(x+sqrt(x^2+1))$
450006	^ACOSH2LNext	( $symb \rightarrow symb'$ )
		$x \rightarrow ln(x+sqrt(x^2-1))$
452006	^ATANH2LNext	$(symb \rightarrow symb')$
		$x \rightarrow \ln((1+x)/(1-x))/2$
454006	^XROOT2ext	( symb1 symb2 $\rightarrow$ symb' )
	_	$x y \rightarrow \exp(\ln(y)/x)$
45A006	^LN2ATAN	$(symb \rightarrow symb')$
		$\mathbf{x} \to \ln(\mathbf{x})$

#### 44.1.3 Simplification, Evaluation and Substitution

Addr.	Name	Description
45B006	^VAR=LIST	( idnt {} → {}' ) Replaces all elements of the initial list by
464006	^SYMBEXEC	<ul> <li>idnt=element.</li> <li>( ob symb → ob' )</li> <li>If symb is an equation, executes the corresponding change of variables in ob, otherwise tries to find symb so that ob is zero. Note that</li> </ul>
465006	^MEVALext	change of variable works for change of user functions.  ( ob {} {}' → ob' )
		Replaces all occurrances of an element of list2 by the corresponding element of list1 in ob. Looks in ob from outer to inner expressions. list2 and list1 may contain secondaries. If
466006	^CASNUMEVAL	vxxlflag is set SIGN var are leaved unchanged. ( symb list1 list2 — symb') Evaluation of a symbolic. The lists' formats are list1={idnt/lam1 idnt_n/lam_n} list2={value1value_n}. The idnt's/lam's in list1 are <i>not</i> evaluated before replacing
467006	^CASCOMPEVAL	value1value_n. ( symb → symb' ) Evaluation of a symbolic.
468006	^REPLACE2BY1	(symb idnt a $\rightarrow$ symb) (symbidnt a $\rightarrow$ symb) (symbidnt a symbolic replacing an idnt by a value; for example evaluation of F(X) for X=1/2)
469006	^NR_REPLACE	( symb idnt a → symb' )  Like REPLACE2BY1 but prevents evaluation of INT.
46B006	^CASCRUNCH	( ob $\rightarrow$ % ) Like CRUNCH but in approximate mode.
46C006	^APPROXCOMPEVAL	( symb → symb') Like CASCOMPEVAL but in approximate mode.
11A007	^ALGCASCOMPEVAL	( expr $ ightarrow$ expr )

Addr.	Name	Description
297006	^SLVARext	( Lvar → Lvar' )
		Simplifies all elements of the list that are sup-
		posed to be variables.
298006	^SIMPLIFY	$( symb \rightarrow symb' )$
		Simplifies one object like EVAL.
299006	^SIMPlext	$(symb \rightarrow symb')$
		Simplifies one object like EXPAND. Object must
		be a symbolic, a real or a complex number.
29A006	^SYMEXPAN	$(symb \rightarrow symb')$
		Simplifies one object like EXPAN. Object must
		be symb/real/cmplx.
29B006	^SIMPVAR	$( ob \rightarrow ob' )$
		Simplifies variable.
2A0006	^SIMPSYMBS	( inf sup fcn var $\rightarrow$
		<pre>int(inf,sup,fcn,var) )</pre>
2A2006	^SIMPUSERFCN	( oblobn #n ob $\rightarrow$ id[] )
		Simplification of user functions. Tests for
		derivative of user functions. Ob must be an
		id, a symbolic, a secondary or a romptr.
2A3006	^EVALUSERFCN	( V1Vn $\#$ n fcn $\rightarrow$ f[])
		Evaluates a user function with stack checking.
2A4006	^SIMP	( ob list $\rightarrow$ ob' )
		Executes the WHERE operator.
2A9006	^SIMPext	$( ob1 ob2 \rightarrow ob1' ob2' )$
		Simplifies two objects in internal representa-
		tion. Checks that o2 is not a complex or an
		irrquad because decomposition of the corre-
		sponding fraction with larg would generate a
	_	"Try to recover Memory".
2AD006	^SIMPGCDext	(o1 o2 gcd $\rightarrow$ o1/gcd o2/gcd)
	_	Divides o1 and o2 by gcd.
2AE006	^SIMP3ext	$(ab \rightarrow ga''b'')$
070005	^= <b>3</b> = <b>3</b> = <b>3</b>	Calculates $g = gcd(a,b)$ and $a''=a/g$ and $b''=b/g$ .
2B9006	^TSIMP2ext	$(\text{symb} \rightarrow \text{symb})$
		Transcendental simplifications. Converts only
		sqrt ^ and XROOT to EXP/LN. LN are returned
		as -1/INV[-LN[]] for use by SERIES.

Addr.	Name	Description
2BA006	^TSIMPext	$( \text{symb} \rightarrow \text{symb} )$
		Transcendental simplifications. Convert transcendental functions to EXP and LN.
2BB006	^TSIMP3ext	( $symb \rightarrow symb$ )

# 44.1.4 Collection and Expansion

Addr.	Name	Description
26E006	^COLCext	( $\operatorname{symb} \to \operatorname{symb}'$ )  Factorization with respect to the current vari-
		able of symb and factorization of the integer
		content of symb.
2FE006	^TCOLLECT	( $symb \rightarrow symb'$ )
		Performs trigonometric linearization and
		then collects sines and cosines of the same an-
	•	gle.
2FF006	^SIGMAEXPext	( symb → symb' )
		Conversion to exp and ln with exponential linearization.
300006	^LINEXPext	nnearization. ( symb → Meta )
300006	LINEXPEXC	Meta = $arg_{exp1}$ coef1 $arg_{expn}$ coefn #2n.
301006	^SIGMAEXP2ext	( Meta → symb )
301000	DIGINALMI ZCAC	Back conversion from arg_exp/coef_meta to
		symbolic.
303006	^SINEXPA	(symb → symb')
		Expands SIN.
316006	^LNEXPA	( $symb \rightarrow symb'$ )
		Expands LN.
31C006	^MTRIG2SYMB	( Meta $\rightarrow$ symb )
		Back conversion of trig-meta to symbolic.
309006	^COSEXPA	( symb → symb' )
200006	^======================================	Expands COS.
30F006	^EXPEXPA	( $symb \rightarrow symb'$ )  Expands EXP.
31B006	^LINEXPA	( symb → Meta )
21000	TINEXEV	Alternates trig operator and coefficient.
31D006	^LNCOLCext	( symb → symb')
		Collects logarithms.
		-

Addr.	Name	Description
31F006	^TEXPAext	( $symb \to symb$ )
		Main transcendental expansion program.
240006	^EXLR	( 'a=b' $\rightarrow$ a b )
		( ob $\rightarrow$ X ob )
		Internal equation splitter.

## 44.1.5 Trigonometric Transformations

Addr.	Name	Description
407006	^HALFTAN	$( \text{symb} \rightarrow \text{symb'} )$
		Converts trigonometric functions to TAN of the
		half angle.
411006	^TRIGTAN	$(symb \rightarrow symb')$
		Convert sin and cos to tan of the same angle.
416006	^TRIGext	$( \text{symb} \rightarrow \text{symb'})$
		Applies sin^2+cos^2=1 to simplify trigonomet-
		ric expressions. If flag -116 is set, tries to keep
		only sin, else only cos.
417006	^HYP2EXPext	$(symb \rightarrow symb')$
		Converts hyperbolic functions to exp and ln.
		Converts XROOT and \(^1\) to exp and \(^1\).
418006	^EXPLNext	$(\text{symb} \rightarrow \text{symb'})$
		Converts all transcendental functions to exp
	_	and ln.
419006	^SERIESEXPLN	$(\text{symb} \rightarrow \text{symb'})$
	_	Converts sqrt, ^ and XROOT to EXP/LN.
426006	^TAN2SCext	$(symb \rightarrow symb')$
	_	Converts tan to sin/cos.
429006	^SIN2TCext	$(symb \rightarrow symb')$
	_	Converts sin to cos*tan.
430006	^ATAN2Sext	$(\text{symb} \rightarrow \text{symb}')$
		Converts ATAN to ASIN using
	<b>^</b>	$asin(x)=atan(x/sqrt(1-x^2)).$
433006	^ASIN2Text	$(\text{symb} \rightarrow \text{symb}')$
		Converts ASIN to ATAN using
		$asin(x)=atan(x/sqrt(1-x^2)).$

Addr.	Name	Description
436006	^ASIN2Cext	( $symb \rightarrow symb'$ )
		Converts ASIN to ACOS using asin(x)=pi/2-
		$a\cos(x)$ .
43A006	^ACOS2Sext	( $symb \rightarrow symb'$ )
		Converts ACOS to ASIN using acos(x)=pi/2-
		asin(x).
43F006	^TAN2SC2ext	( $symb \rightarrow symb'$ )
		Converts TAN to SIN/COS of the double angle.
		If flag -116 is set calls TAN2SC2, else TAN2CS2.
456006	^LN2ext	( $symb \rightarrow symb'$ )
		If symb contains x, returns $-1/\text{inv}(-\ln(x))$ , else
		ln(x). Used by SERIES.

# 44.1.6 Division, GCD and LCM

Name	Description
^PSEUDODIV	( Q2 Q1 $\rightarrow$ a Q2*a/Q1 Q2*a/Q1 )
^BESTDIV2	( o2 o1 $ ightarrow$ quo mod )
^QUOText	( o2 o1 $\rightarrow$ o2 div o1 )
	Euclidean quotient of 2 objets (works even if o2
	mod o1=0).
^NEWDIVext	( ob2 ob1 $ ightarrow$ quo mod )
	Euclidean division, ob2 and ob1 may be frac-
	tions of returns a fraction of Q.
^QUOTOBJext	( a_a-1a0 bb_1b0 #b #a flag $ ightarrow$
	rq)
	SRPL Euclidean division: step 2 computes the
	remainder r only if flag is TRUE.
^DIVISIBLE?	( a b $\rightarrow$ a/b T )
	( a b $\rightarrow$ ob F )
	Returns TRUE and quotient if b divides a, oth-
	erwise returns FALSE.
^QDiv?	( $a b \rightarrow a/b T$ )
	( a b $\rightarrow$ F )
	Returns TRUE and quotient if b divides a, oth-
	erwise returns FALSE.
	^PSEUDODIV ^BESTDIV2 ^QUOTEXT  ^NEWDIVEXT  ^QUOTOBJEXT  ^DIVISIBLE?

Addr.	Name	Description
3F6006	^FastDiv?	( P Q → P/Q PmodQ T ) Euclidean division. Assumes P and Q have integer or Gaussian integer coefficient. Returns FALSE in complex mode or if sparse short division fails.
3F7006	^POTENCEext	( z1 z2 $\rightarrow$ q r ) Step by step Euclidean division for small integers.
2A5006	^DENOLCMext	( list → ob ) Calculates the LCM of the denominator of the elements of the list. If input is not a list, returns the denominator of the object.
2A6006	^METADENOLCM	( Meta $\rightarrow$ ob ) Calculates LCM of the denominators of the elements of Meta.
2B1006	^LPGCDext	( $\{\} \rightarrow \{\} \text{ ob })$ Calculates the GCD of all the elements in the list. The algorithm is far from optimal.
2B2006	^SLOWGCDext	(class) contains irrquads. cisthe GCD of the contents of the original polynomials returned after failure of GCDHEUext.
2B3006	^QGcd	( ob2 ob1 $\rightarrow$ gcd ) Generic internal GCD. ( LAM2: GCDext ob1, ob2 $\rightarrow$ pgcd ).

# **Chapter 45 Symbolic Meta Handling**

This chapter contains words that modify metas which are exploded symbolic objects. They are used to modify the expression or to operate on them.

#### 45.1 Reference

#### 45.1.1 Basic Expression Manipulation

Addr.	Name	Description
157006	^SYMBINCOMP	( symb $\rightarrow$ ob1 obN $\#$ n )
		( ob $ ightarrow$ ob #1 )
		( $\{\} \rightarrow \{\} \#1 $ )
		Explodes symbolic object into meta. Other ob-
		jects are converted into one-object metas by
		pushing #1 into the stack.
386006	m-1&m+1	( meta $ ightarrow$ meta&1&+ meta&1&- )
		Creates two copies of the meta. To the first one,
		adds 1 and +, to the second one, adds 1 and
387006	^meta1/meta	, , , ,
		Duplicates the meta, and inverts the expression
		represented by it.
388006	^1&meta	( Meta $ ightarrow$ 1&Meta )
		Prepends the number 1 to the meta.
389006	^meta/2	$($ Meta $\rightarrow$ Meta $\&2\&/$ $)$
		Divides the expression by two.
38A006	^addt2	( Meta $ ightarrow$ Meta $\&2$ )
		Appends the number 2 to the meta.
38B006	^addt/	( Meta → Meta&/ )
		Appends division to meta.

Addr.	Name	Description
38C006	^meta2*	( Meta $ ightarrow$ 2&Meta&* )
		Multiplies the expression by 2.
459006	^metai*	( $meta \rightarrow meta*i$ )
		Multiplies meta by i.
38D006	^meta1-sq	( Meta $ ightarrow$ 1&Meta&SQ&- )
		Changes x into 1-x <sup>2</sup> , where x is the original ex-
		pression.
38E006	^metasq+1	( Meta $ ightarrow$ Meta&SQ&1&+ )
		Changes x into $x^2+1$ , where x is the original ex-
		pression.
38F006	^metasq-1	( Meta $ ightarrow$ Meta&SQ&1&- )
		Changes x into $x^2-1$ , where x is the original
		equation.
390006	^meta-1	( Meta $ ightarrow$ Meta&1&- )
		Subtracts one from the expression.
398006	^addt^	( Meat $\rightarrow$ Meta $\&$ ^ )
		Append power operator to meta object.
39C006	^top&addt*	( meta2 meta1 $ ightarrow$ meta2*meta1 )
		top& addt*. No checks.
39D006	^top&addt/	( meta2 meta1 $ ightarrow$ meta2/meta1 )
		top& addt/. No checks.
39E006	^addti	( $meta  ightarrow meta\&i$ )
		Appends i (the Imaginary unit) to expression.

# 45.1.2 Basic Operations and Function Application

Addr.	Name	Description
393006	^metaadd	( Metal Meta2 $ ightarrow$ Metal+Meta2 )
		Adds 2 meta objects with trivial simplifica-
		tions. metaadd checks for Meta1/2=Z0 ONE.
3AB006	^MetaAdd	( Meta2 Meta1 $ ightarrow$ Meta2+Meta1 )
		Adds 2 meta objects with trivial simplifica-
		tions. Checks for infinities then call metaadd.
1CE006	^ckaddt+	( Metal Meta2 $ ightarrow$ Metal+Meta2 )
		Adds 2 meta objects with trivial simplifica-
		tions.

Addr.	Name	Description
394006	^metasub	( Metal Meta2 → Metal+Meta2 ) Subtracts 2 meta objects with trivial simplifi- cations. metasub checks for Metal/2=Z0 ONE.
3AD006	^MetaSub	( Meta2 Meta1 $\rightarrow$ Meta2-Meta1 ) Subtracts 2 meta objects with trivial simplifications. Checks for infinities then call metasub.
1CF006	^ckaddt-	( Metal Meta2 $\rightarrow$ Metal+Meta2 ) Subtracts 2 meta objects with trivial simplifications.
395006	^metamult	( Meta1 Meta2 $\rightarrow$ Meta1*Meta2 ) Multiplies 2 meta objects with trivial simplifications. Checks for meta1, meta2= Z0 or Z1, checks for xNEG.
3AF006	^MetaMul	( Meta2 Meta1 $\rightarrow$ Meta2*Meta1 ) Multiplies 2 meta objects with trivial simplifications. Checks for infinities/0 then call metamult.
1CD006	^ckaddt*	( Metal Meta2 $\rightarrow$ Metal*Meta2 ) Multiplies 2 meta objects with trivial simplifications.
396006	^metadiv	( Meta2 Meta1 $\rightarrow$ Meta2/Meta1 ) Divides 2 meta objects with trivial simplifications. Checks for infinities and 0, meta2 =1 or Z-1, checks for xNEG.
3B1006	^MetaDiv	( Meta2 Meta1 $\rightarrow$ Meta2/Meta1 ) Divide 2 meta objects with trivial simplifications. Checks for infinities and 0 then call metadiv.
3F1006	^DIVMETAOBJ	( o1on $\#n$ ob $\rightarrow$ {o1/obon/ob} ) Division of all elements of a meta by ob. Tests if o=1.
397006	^meta^	( Meta ob $\rightarrow$ Meta&ob&^ ) Elevates expression to a power. If ob=1, just returns the expression. Tests for present of xNEG in the end of meta for integral powers.

Addr.	Name	Description
399006	^metapow	( Meta2 Meta1 → Meta2^Meta1 )
		Elevates expression to a power (any other ex-
		pression). If length of Meta1 is ONE, calls
		meta^.
3B5006	^MetaPow	( Meta2 Meta1 $\rightarrow$ Meta2^Meta1 )
		Power. Checks for infinities then calls
		metapow.
39B006	^metaxroot	( Meta2 Meta1 $\rightarrow$ Meta2&XROOT&Meta1 )
		Root of expression.
3B9006	^metaneg	$($ meta $\rightarrow$ meta $)$
	_	Checks only for meta finishing by xNEG.
3BA006	^metackneg	( meta $\rightarrow$ meta )
	_	Like metaneg but checks for meta=ob ONE.
3B7006	^MetaNeg	( Meta → Meta )
	J	Negates meta. Only checks for metas finish-
		ing by xNEG.
502006	^xSYMRE	( meta → meta' )
		Meta complex real part. Expands only + - */
		^.
504006	^xSYMIM	( $meta \rightarrow meta'$ )
		Meta complex imaginary part. Expands only
		+-*/^.
50E006	^addtABS	( Meta $ ightarrow$ Meta' )
		Meta ABS. Does a CRUNCH first to find sign.
510006	^addtABSEXACT	( Meta $ ightarrow$ Meta' )
		Meta ABS. No crunch, sign is only found us-
		ing exact methods.
511006	^addtSIGN	( Meta $ ightarrow$ Meta' )
		Meta SIGN.
513006	^addtARG	( Meta $ ightarrow$ Meta' )
		Meta ARG.
12D006	^addtXROOT	( Meta2 Meta1 $ ightarrow$ Meta' )
		Meta XROOT. $XROOT(o2,o1)$ is $o1^{1/o2}$ ,
		compared to o2^o1.
12F006	^addtMIN	( Meta2 Meta1 $\rightarrow$ Meta' )
		Meta MIN.
131006	^addtMAX	( Meta2 Meta1 → Meta' )
		Meta MAX.

Addr.	Name	Description
133006	^addt<	( Meta2 Meta1 → Meta' )
105006	A 17.	Meta <.
135006	^addt<=	( Meta2 Meta1 $ ightarrow$ Meta' )
127006	^addt>	Meta <=. ( Meta2 Meta1 $ ightarrow$ Meta' )
13/000	addt/	Meta > Meta > Meta
139006	^addt>=	( Meta2 Meta1 $\rightarrow$ Meta' )
137000	addc/-	Meta >=.
13B006	^addt==	( Meta2 Meta1 → Meta' )
		Meta ==.
13D006	^addt!=	( Meta2 Meta1 $ ightarrow$ Meta' )
		Meta !=.
13F006	^addt%	( Meta2 Meta1 $ ightarrow$ Meta' )
		Meta %.
141006	^addt%CH	( Meta2 Meta1 $ ightarrow$ Meta' )
		Meta %CH. Meta $2*(1+Meta'/100)=Meta1$ .
143006	^addt%T	( Meta2 Meta1 → Meta' )
		Meta %T.
145006	^addtMOD	( Meta2 Meta1 → Meta' )
1 47006	^ - 11+mpag	Meta MOD.
14/006	^addtTRNC	( Meta2 Meta1 $ ightarrow$ Meta' ) Meta TRNC.
149006	^addtRND	( Meta2 Meta1 $\rightarrow$ Meta' )
117000	addelinb	Meta RND.
14B006	^addtCOMB	( Meta2 Meta1 → Meta' )
		Meta COMB.
14D006	^addtPERM	( Meta2 Meta1 $ ightarrow$ Meta' )
		Meta PERM.
14F006	^addtOR	( Meta2 Meta1 $ ightarrow$ Meta' )
		Meta OR.
151006	^addtAND	( Meta2 Meta1 $ ightarrow$ Meta' )
		Meta AND.
153006	^addtXOR	( Meta2 Meta1 → Meta' )
F06006	^ - 11+ 00377	Meta XOR.
506006	^addtCONJ	( meta → meta' )  Moto compley conjugate
523006	^addtLN	Meta complex conjugate. ( Meta → Meta' )
323000	auutuN	Meta → Meta )  Meta LN.
		Micua III.

Addr.	Name	Description
535006	^addtCOS	( Meta → Meta' )
525006	^ 11. g = 1.	Meta COS.
537006	^addtSIN	( Meta → Meta' )
		Meta SIN.
539006	^addtTAN	( Meta → Meta' )
		Meta TAN.
53B006	^addtSINACOS	( meta → meta' )
		If meta stands for x, meta' stands for sqrt[1-x^2].
E30006	^addtASIN	( Meta → Meta' )
330000	addlasin	$($ Meta $\rightarrow$ Meta $)$ Meta ASIN.
F277006	^- 44+7,000	Meta ASIN. ( Meta → Meta' )
53E006	^addtACOS	$(\text{Meta} \rightarrow \text{Meta})$ Meta ACOS.
540006	^addtATAN	Meta ACOS. ( Meta → Meta' )
340000	addtalan	( Meta → Meta ) Meta ATAN.
542006	^addtSINH	( Meta → Meta' )
342000	addtSINH	Meta SINH.
544006	^addtCOSH	( Meta → Meta' )
344000	addccosii	Meta COSH.
546006	^addtTANH	( Meta → Meta' )
01000	0.0.0.0 11 11.11	Meta TANH.
549006	^addtATANH	( Meta → Meta' )
		Meta ATANH.
54C006	^addtASINH	( Meta $ ightarrow$ Meta' )
		Meta ASINH.
54F006	^addtACOSH	( Meta $ ightarrow$ Meta' )
		Meta ACOSH.
551006	^addtSQRT	( Meta $ ightarrow$ Meta' )
		Meta SQRT.
554006	^addtSQ	( Meta $ ightarrow$ Meta' )
		Meta SQ.
556006	^addtINV	( Meta $ ightarrow$ Meta' )
		Meta INV.
558006	^addtEXP	( Meta $ ightarrow$ Meta' )
		Meta EXP. Does not apply EXP[-
		]=1/EXP[].
559006	^xSYMEXP	( Meta $ ightarrow$ Meta' )
		Meta EXP. Applies EXP[]=1/EXP[].

Addr.	Name	Description
55A006	^addtD->R	( Meta $ ightarrow$ Meta' ) Meta $D  ightarrow R$ .
55C006	^addtR->D	( Meta $ ightarrow$ Meta' ) Meta $R{ ightarrow}D.$
55E006	^addtFLOOR	( Meta $ ightarrow$ Meta' ) Meta $ ext{FLOOR}.$
560006	^addtCEIL	( Meta $ ightarrow$ Meta' ) Meta CEIL.
562006	^addtIP	( Meta $ ightarrow$ Meta' ) Meta IP.
564006	^addtFP	( Meta $ ightarrow$ Meta' ) ${f Meta~FP}.$
566006	^addtXPON	( Meta → Meta' ) Meta XPON.
568006	^addtMANT	( Meta → Meta' ) Meta MANT.
56A006	^addtLNP1	( meta $ ightarrow$ meta ) Meta LNP1.
56C006	^addtLOG	( meta $ ightarrow$ meta ) Meta LOG.
56E006	^addtALOG	( meta → meta ) Meta ALOG.
570006	^addtEXPM	( meta → meta ) Meta EXPM.
574006	^addtFACT	( Meta → Meta' ) Meta FACT.
577006	^addtNOT	( Meta → Meta' ) Meta NOT.

## **45.1.3** Trigonometric and Exponential Operators

Addr.	Name	Description
409006	^cos2tan/2	( meta $ ightarrow$ meta' )
		$x \to (1-(\tan(x/2))^2)/(1+(\tan(x/2))^2)$
40A006	$1-x^2/1+x^2$	( meta $ ightarrow$ meta' )
		$\mathbf{x} \rightarrow (1\text{-}\mathbf{x}^2)/(1\text{+}\mathbf{x}^2)$
40C006	^sin2tan/2	( $meta \rightarrow meta'$ )
		$x \rightarrow 2 \tan(x/2)/(1+(\tan(x/2))^2)$

Addr.	Name	Description
40D006	^2x/1+x^2	( meta $ ightarrow$ meta' )
		$ ext{x}  ightarrow 2 ext{x}/(1+ ext{x}^2)$
40F006	^tan2tan/2	( meta $ ightarrow$ meta' )
		$x \rightarrow 2 \tan(x/2)/(1-(\tan(x/2))^2)$
410006	^addtTAN/2	( meta $ ightarrow$ meta' )
		$x \rightarrow tan(x/2)$
413006	^cos2tan	( meta $ ightarrow$ meta' )
		$x \rightarrow 1/sqrt(1+(tan(x))^2)$
415006	^sin2tan	( meta $ ightarrow$ meta' )
		$x \rightarrow tan(x)/sqrt(1+(tan(x))^2)$
421006	^tan2exp	( meta $ ightarrow$ meta' )
		$x \rightarrow (\exp(i2x)\text{-}1)/(i^*(\exp(i2x)\text{+}1))$
423006	^asin2ln	( meta $ ightarrow$ meta' )
		$x \rightarrow = i*ln(x+sqrt(x^2-1))+\pi/2.$
425006	^acos2ln	( meta $ ightarrow$ meta' )
		$x \rightarrow ln(x+sqrt(x^2-1))/i$
428006	^sin/cos	( $meta \rightarrow meta'$ )
		$x \rightarrow \sin(x)/\cos(x)$
42B006	^cos*tan	( $meta \rightarrow meta'$ )
		$x \rightarrow \cos(x) * \tan(x)$
42D006	^sqrt1-sin^2	( meta $ ightarrow$ meta' )
		$x \rightarrow sqrt(1-(sin(x))^2).$
42F006	^sqrt1-cos^2	( $meta \rightarrow meta'$ )
		$x \rightarrow sqrt(1-(cos(x))^2).$
432006	^atan2asin	( meta $ ightarrow$ meta' )
		$x \rightarrow asin(x/sqrt(x^2+1))$
435006	^asin2atan	( meta $ ightarrow$ meta' )
		$x \rightarrow atan(x/sqrt(1-x^2))$
438006	^pi/2-acos	( $meta \rightarrow meta'$ )
		$x \to \pi/2$ -acos(x)
439006	^pi/2-meta	( $meta \rightarrow meta'$ )
		${f x}  ightarrow \pi/2$ - ${f x}$
43B006	^pi/2-asin	( meta → meta' )
10-00-	0.7	$x \rightarrow \pi/2$ -asin(x)
43E006	^atan2ln	( meta → meta' )
441006	0041	$ ext{x}  ightarrow  ext{i/2*ln((i+x)/(i-x))}$
441006	^2*1-cos/sin	$(\text{meta} \rightarrow \text{meta'})$
		$x \rightarrow (1-\cos(2x))/\sin(2x)$

Addr.	Name	Description
443006	^2*sin/1+cos	( meta $ ightarrow$ meta' )
		$x \rightarrow \sin(2x)/(1+\cos(2x))$
445006	^sin2exp	( $meta \rightarrow meta'$ )
		$x \rightarrow (e^{(i*x)-1/e^{(i*x))/(2i)}$
447006	^cos2exp	( $meta \rightarrow meta'$ )
		$x \rightarrow (e^(i*x)+1/e^(i*x))/2$
449006	^sinh2exp	( $meta \rightarrow meta'$ )
		$x \rightarrow (e^x-1/e^x)/2$
44B006	^cosh2exp	( $meta \rightarrow meta'$ )
		$x \rightarrow (e^x+1/e^x)/2$
44D006	^tanh2exp	( $meta \rightarrow meta'$ )
		$x \rightarrow (e^2x-1)/(e^2x+1)$
44F006	^asinh2ln	( $meta \rightarrow meta'$ )
		$x \rightarrow ln(x+sqrt(x^2+1))$
451006	^acosh2ln	( $meta \rightarrow meta'$ )
		$x \rightarrow ln(x+sqrt(x^2-1))$
453006	^atanh2ln	( meta $ ightarrow$ meta' )
		$x \rightarrow \ln((1+x)/(1-x))/2$
455006	^xroot2expln	( meta1 meta2 $ ightarrow$ meta' )
		$x y \rightarrow \exp(\ln(y)/x)$
458006	^exp2sincos	( $meta \rightarrow meta'$ )
		Returns EXP of meta as
		EXP[RE]*[COS+i*SIN].

#### 45.1.4 Infinity and Undefs

Addr.	Name	Description
3A1006	^1metaundef#	( meta $ ightarrow$ meta $\#$ )
		Tests presence of undef in meta. # is the posi-
		tion of undef.
3A0006	^2metaundef#	( meta2 meta1 $ ightarrow$ meta2 meta1 $\sharp$ )
		Tests presence of undef in meta2 and meta1.
		# is the position of undef.
3A2006	^metaundef	( $ ightarrow$ meta )
		Returns undef meta.
3A4006	^1metainf#	( meta $ ightarrow$ meta $\#$ )
		Finds position of infinity in meta. Metas of
		length>2 are considered as finite meta.

Addr.	Name	Description
3A3006	^2metainf#	( meta2 meta1 $ ightarrow$ meta2 meta1 $\sharp$ )
		Finds position of infinity in meta 2 and
		meta1. Metas of length>2 are considered as
		finite meta.
3A5006	^metainftype	( meta $ ightarrow$ # )
		Returns infinity type: 1 for +infinity, 2 for -
		infinity or 0 for unsigned.
3A6006	^unsignedinf	( $ ightarrow$ meta )
		Returns unsigned infinty.
3A7006	^plusinf	( $ ightarrow$ meta )
		Returns plus infinty.
3A8006	^NDROPplusinf	( oblobn $ ightarrow$ meta )
		Replaces meta by plus infinty.
3A9006	^minusinf	( $ ightarrow$ meta )
		Returns minus infinty.
3AA006	^NDROPminusinf	( oblobn $ ightarrow$ meta )
		Replace meta by minus infinty.

# 45.1.5 Expansion and Simplification

Addr.	Name	Description
3BB006	^metasimp	( Meta $ ightarrow$ Meta )
		Simplifies a meta object. Non recursive ratio-
		nal simplification.
118007	^DISTRIB*	( meta $ ightarrow$ meta' T )
		( meta $ ightarrow$ meta F )
		Distribute *. Returns FALSE if no distribution
		done.
3C2006	^DISTRIB/	( meta $ ightarrow$ meta' T )
		( meta $ ightarrow$ meta F )
		Distribute /. Returns FALSE if no distribution
		done.
304006	^METASINEXPA	( Meta $ ightarrow$ Meta' )
		Expands SIN.
305006	^SINEXPA+	( Meta $ ightarrow$ Meta' )
		Expands $SIN(x+y)$ .
306006	^SINEXPA-	( Meta $ ightarrow$ Meta' )
		Expands SIN(x-y).

Addr.	Name	Description
307006	^SINEXPA*	( Meta → Meta' )
		Expands $SIN(x*y)$ . Expands if x or y is an in-
		teger.
308006	^SINEXPA*1	( Meta2 Meta1 $ ightarrow$ Meta' )
		Expands SIN(x*y). Meta1 is assumed to be an
		integer.
30A006	^METACOSEXPA	( Meta $ ightarrow$ Meta' )
		Expands COS.
30B006	^COSEXPA+	( Meta $ ightarrow$ Meta' )
		Expands $COS(x+y)$ .
30C006	^COSEXPA-	( Meta $ ightarrow$ Meta' )
		Expands $COS(x-y)$ .
30D006	^COSEXPA*	( Meta $ ightarrow$ Meta' )
		Expands $COS(x*y)$ .
30E006	^COSEXPA*1	( $meta2 \ meta1 \rightarrow Meta'$ )
		Expands COS(x*y). meta1 represents an inte-
		ger.
310006	^METAEXPEXPA	( Meta $ ightarrow$ Meta' )
		Expands EXP.
311006	^EXPEXPA+	( Meta $ ightarrow$ Meta' )
		Expands $EXP(x+y)$ .
312006	^EXPEXPA-	( Meta $ ightarrow$ Meta' )
		Expands EXP(x-y).
313006	^EXPEXPA*	( Meta $\rightarrow$ Meta' )
		Expands EXP(x*y).
314006	^EXPEXPANEG	( Meta → Meta' )
		Expands EXP(-x).
315006	^EXPEXPA*1	( Meta2 meta1 $\rightarrow$ Meta' )
		Expands EXP(x*y). meta1 represents an inte-
		ger.
317006	^METALNEXPA	( Meta → Meta' )
		Expands LN.
318006	^LNEXPA*	( Meta → Meta' )
		Expands LN(x*y).
319006	^LNEXPA/	( Meta → Meta' )
		Expands LN(x/y).
31A006	^LNEXPA^	( Meta → Meta' )
		Expands $LN(x^y)$ .

Addr.	Name	Description
31E006	^METATANEXPA	( meta → tan[meta] )
		Expands tan[meta].

#### 45.1.6 Tests

Addr.	Name	Description
39A006	^metafraction?	( Meta $ ightarrow$ Meta flag )
		Tests if meta is a fraction of integers.
3BC006	^metapi?	( Meta $ ightarrow$ Meta $\sharp$ )
		Tests presence of $\pi$ in a meta. # is the last
		occurrence of $\pi$ or 0.
3BD006	^metaCOMPARE	( Meta2 Meta1 $ ightarrow$ Meta2 Meta1 $\#$ )
		Comparison of 2 meta.
		# =0 if undef
		# =1 if >
		# =2 if <
		# =3 if =
		Assumes generic situation, e.g. X <sup>2</sup>
		> 0 in real mode. Look below
		STRICTmetaCOMPARE for a more care-
		ful comparison.
3BE006	^STRICTmetaCOMPARE	( Meta2 Meta1 $ ightarrow$ Meta2 Meta1 $\#$ )
		Comparison of 2 meta.
		# =0 if undef
		# =1 if >
		# =2 if <
		# =3 if =
		Unlike metaCOMPARE it does not assume
		generic situation.
3C3006	^metareal?	( meta $ ightarrow$ meta flag )
		Tests if $IM[meta]==0$ .

# **Chapter 46 Polynomials**

The entries in this chapter deal with computation with Polynomials.

#### 46.1 Reference

#### **46.1.1 Computation with Polynomials**

Addr.	Name	Description
118006	^QAdd	$( o1 \rightarrow o2+o1 )$
		Adds two polynomials.
119006	^RADDext	( o2 o1 $\rightarrow$ o2+o1 )
		Internal +. This is the same entry as ^QAdd.
117006	^SWAPRADD	( o2 o1 $\rightarrow$ o1+o2 )
		SWAP, then QAdd.
115006	^QSub	( o2 o1 $ ightarrow$ o2-o1 )
		Subtracts two polynomials.
116006	^RSUBext	( o2 o1 $\rightarrow$ o2-o1 )
		Internal This is the same entry as ^QSub.
114006	^SWAPRSUB	( o2 o1 $\rightarrow$ o1-o2 )
		SWAP, then QSub.
111006	^QMul	( Q1 Q2 $ ightarrow$ Q )
		Multiplication of polynomials with extensions.
112006	^RMULText	( Q1 Q2 $ ightarrow$ Q )
		Multiplication of polynomials with extensions.
		This is the same entry as ^QMul.
110006	^SWAPRMULT	( Q1 Q2 $ ightarrow$ Q )
		SWAP, then ^QMul.
11C006	^QDiv	( o2 o1 $\rightarrow$ o2/o1 )
		Internal /.

372 46. Polynomials

Addr.	Name	Description
11B006	^RDIVext	( o2 o1 $\rightarrow$ o2/o1 )
		Internal /. This is the same entry as ^QDiv.
11A006	^SWAPRDIV	( o2 o1 $\rightarrow$ o1/o2 )
		SWAP, then QDiv.
0D9006	^QMod	( Q, $Z \rightarrow Q \mod Z$ )
113006	^RASOP	( $n1/d1$ $n2/d2 \rightarrow d1*d2$ $n1*d2$ $n2*d1$ ) Used by RADDext and RSUBext for rational in-
11-006	A = = !!	put.
11F006	^RP#	$( \circ 2 \# \rightarrow \circ 2^{\uparrow} \# )$
10000	A150	Internal power (not for matrices).
120006	^MPext	( ob # prg* → ob^# )
		General power with a specified multiplication
102006	^==	program.
123006	^RPext	( o2 o1 $\rightarrow$ o2^o1 )
		Tries to convert o1 to an integer to call RP#, otherwise x^ext.
100006	^DISTDIVext	
108006	DISTDIVEXU	$(PQ \rightarrow quo mod T)$ $(PQ \rightarrow PQF)$
		$PQ \rightarrow PQP$ Euclidean division. Assumes P and Q have in-
		teger coefficientes. Returns FALSE if sparse
		short division fails.
355006	^PTAYLext	( P, $r \rightarrow \text{symb}$ )
363000	FIAILCAC	Taylor for polynomials.
15B006	^CARCOMPext	( $Q1/Q2 \rightarrow Q1'/Q2'$ )
130000	CHICOMI CKC	Extracts leading coefficients for the first vari-
		able from a rational polynomial.
3EE006	^QDivRem	( ob2 ob1 $\rightarrow$ quo mod )
311000	QDIVICIII	Polynomial Euclidean division of 2 objects.
		Dispatchs to DIV2LISText for list polynomi-
		als.
3EF006	^DIV2LISText	$($ Z0 11 12 $\rightarrow$ div mod $)$
		Euclidean division, l1 and l2 are list polynomi-
		als. Test first if l1=l2, then tries fast division,
		if it fails switch to SRPL division.
3F8006	^PDIV2ext	( A B $\rightarrow$ Q R )
		Step by step Euclidean division for univar poly.
3F9006	^PSetSign	( P1 P2 $\rightarrow$ sign[P2]*P1 )
		Sets sign of P1 according to leading coeff of P2.
3C4006	^ModExpa	( ${\tt Zn\ Fraction} \to {\tt Fraction\ modulo\ Zn}$ )

Addr.	Name	Description
3C5006	^ModAdd	( Q1 Q2 Zn $ ightarrow$ Z )
		Modular addition. $Z = Q1+Q2 \pmod{Zn}$ .
3C6006	^ModSub	( Q1 Q2 $Zn \rightarrow Z$ )
		Modular subtraction. $Z = Q1-Q2 \pmod{Zn}$ .
3C7006	^ModMul	( Q1 Q2 $Zn \rightarrow Z$ )
		Modular multiplication. $Z = Q1*Q2 \pmod{Zn}$ .
3C8006	^ModDiv	( Z1 Z2 Zn $ ightarrow$ Z )
		Modular division. $Z = Z1/Z2 \pmod{Zn}$ .
3C9006	^ModDiv2	( Q1 Q2 Zn $ ightarrow$ quo mod mod' )
		Modular division. mod' = Q1 mod Q2 mod Zn.
		If Q1 and Q2 are integers, Q1 mod Q2 mod Zn
		is always 0.
3CA006	^ModInv	( $Z Zn \rightarrow Z'$ )
		Modular inversion. $Z' = INV(Z) \pmod{Zn}$ .
		NONINTERR if $GCD[Z,Zn]  eq 1$ or if $Z$ = $0$ (other-
		wise the results would be unpredictable).
3CB006	^ModGcd	( Q1 Q2 Zn $\rightarrow$ Q' )
		Modular GCD.

### 46.1.2 Factorization

Addr.	Name	Description
08E006	^BerlekampP	( P $\#$ prime $\rightarrow$ P F / P Lf $\#$ prime T ) Berlekamp's algorithm for finding modular fac-
		tors of a univariate polynomial.
08F006	^Berlekamp	( P → P F / P Lf #prime T ) Berlekamp's algorithm for finding modular factors of a univariate polynomial with a leading frontend for finding linear factors faster. The input polynomial must be square free, otherwise the polynomial is not fully factored. Due to memory restrictions byte sized coefficients are used and the following restrictions were imposed: prime<128 and degree<256. If the conditions are not met FALSE is returned. BCD: prime≤97.

374 46. Polynomials

Addr.	Name	Description
0A8006	^ALG48FCTR?	( P $\rightarrow$ [ meta cst_coeff TRUE   P FALSE ] ) Factorizes square-free polynomial in Erable format.
0A9006	^MFactTriv	( $P \rightarrow meta-factor P'$ ) Extracts all trivial power factors of P.
0AA006	^CheckPNoExt	( $P \rightarrow P$ flag ) Checks that P does not contain any DOCOL (i.e. extensions).
0AB006	^PPP	( P $\rightarrow$ PP PC ) Computes primitive polynomial and content of non-const P with respect to X1. The results are trimmed (provided P was).
0AC006	^PFactor	( $P \rightarrow Lfk Z$ ) Does a complete factorization of P. The result is trimmed.
0AD006	^PSqff	( P → Lfk ) Square-free and trivial factorization, including integer content, of P taken positive. Factors of same power are not necessarily merged or adjacent, but all Fi's are square-free.
0AE006	^PHFctr	(P → Lf)  Heuristic factorization of polynomial taken positive. LAM FullFact? must be bound. If LAM FullFact? is TRUE, a full factorization is done. If it is FALSE, only square-free and trivial factorization is done.
0AF006	^PHFctr1	( P → Lf ) Heuristic factorization of primitive polynomial. LAM FullFact? must be bound. If TRUE, a full factorization is done. When FALSE, only a square-free and trivial factorization are done.
0B0006	^PHFctr0	( $P \rightarrow Lf$ ) Heuristic factorization of primitive square-free non constant polynomial.
0D8007	^P2P#	( $P \rightarrow P' \#$ ) Extracts trivial power of poly. P must be a valid poly (if list, begin with a non zero coeff).

Addr.	Name	Description
0B1006	^DeCntMulti	( R $\rightarrow$ L ) Transforms list with count into simple list. R = { $\{f1 \#k1\} \dots \{fn \#kn\} \}$ L = { $f1 f1 \dots fn fn \}$ .
0B2006	^DoLS	( $L S F \rightarrow L'$ ) Applies program F(Li,S) to every elem of L.
0B3006	^PNFctr	( $Z \rightarrow Lf$ ) Factorization of positive integer as polynomial. Lf = {} if Z is 1 Lf = { $Z1 \#k1$ } $Zn \#kn$ } o/w.
0B4006	^PSQFF	( P $\rightarrow$ Lsqff ) Computes the square-free factorization of primitive P. The result is trimmed (provided P was).
0B5006	^LiftZAdic	( p z F $\rightarrow$ L ) Lift n-1 z-adic factorization into n factorization.
0B6006	^LFCProd	$(\ \ C\ \ L\  o\ C\ \ P\ )$ Calculates combination product.
0B7006	^UFactor	( $P \rightarrow Lf$ ) Factorization of a square free primitive univariate polynomial.
0B8006	^UFactor1	( $P \rightarrow Lf$ ) Factorization of a square free primitive univariate polynomial of degree $> 2$ .
0В9006	^MonicLf	( Lfp p $\rightarrow$ Lfp' ) Converts true modular factorization to monic factorization by dividing by the leading coefficient of factor 1.
0BA006	^DemonicLf	( Lfp lc p $\rightarrow$ Lfp' ) Converts monic modular factorization to true modular factorization by multiplying factor1 by lcoeff.

376 46. Polynomials

Addr.	Name	Description
0BB006	^LiftLinear	<pre>( #root1 #rootn #n → ) Lifts modular roots of a polynomial to find linear factors of a univariate polynomial. Lflin = list of found true factors Lfplin' = remaining linear factors P' = remaining polynomial Assumes UFactor lambda variables available</pre>
0BC006	^LiftGeneral	and uses them for input and output.  ( → )  Lifts factorization mod p to factorization mod p^k where p^k exceeds the factor bound for successful true factor extraction. Assumes UFactor lambda variables.
0BD006	^UFactorDeg2	( $P \rightarrow Lf$ ) Factorization of a degree 2 polynomial. Polynomial is univariate, square free and primitive.
0BE006	^CombineFac	( P Lfp p $\rightarrow$ Tf Tfp ) Combines modular factors to true factors. P is the polynomial to factor, Lfp is the list of mod- ular factors, and p the modulo. The entry re- turns the a list of found true factors (Tf) and the list of modular factors for each true factor (Tfp)
0BF006	^CombProd	( lc Lfp p Cb $\rightarrow$ F )  Calculates modular combination.
0C0006	^CombInit	( $\#r \rightarrow Cb$ )  Inits modular combination list to value { 1 0 0 0 }.
0C1006	^CombNext	( Cb → Cb' flag ) Gets next possible modular combination. Assumes Cb is valid and is in tempob area.
0C2006	^RmCombNext	( Lf Cb → Lfrm Lf' Cb' flag ) Removes next possible combination after a successful combination has been found, and remove the used factors from the factor list.
0C3006	^PFactTriv	( $P \rightarrow P'$ Lf ) Extracts all trivial power factors of P.

Addr.	Name	Description
0C4006	^VarFactor	( P #var → P #n )
		Calculates what power of the given variable is
		a factor in P.
0C5006	^PFactPowCnt	( P $ ightarrow$ P Lk flag )
		Calculates trivial power factors in P. flag is
		TRUE if any of the powers is nonzero.
0C6006	^PDivLk	( P $Lk \rightarrow P'$ )
		Divides polynomial by its trivial powers.
282006	^FEVIDENText	( P $ ightarrow$ meta-fact cst coeff )
		Real mode: full factorization over the integer
		Complex mode: find all 1st order factors of P.

# 46.1.3 General Polynomial Operations

Addr.	Name	Description
09B006	^ONE{}POLY	( ob $\rightarrow$ {ob} ob1 $\rightarrow$ Q )
		Replaces ONE { }N for polynomial building.
09C006	^TWO{}POLY	( ob1 ob2 $\rightarrow$ Q )
		Replaces TWO $\{\}$ N for polynomial building.
09D006	^THREE{}POLY	( ob1 ob2 ob3 $\rightarrow$ Q )
		Replaces THREE { }N for polynomial building.
09E006	^TWO::POLY	( ob1 ob2 $\rightarrow$ :: )
		Replaces 20b>Seco for polynomial building.
09F006	^::POLY	( Meta $ ightarrow$ :: )
		Replaces :: N for polynomial building. As op-
		posed to the regular :: N code, we do pop the
		binary number. This is enforced by the entry
		to the common polyxml code.
0A0006	^{}POLY	( Meta $ ightarrow$ Q )
		Replaces {}N for polynomial building. As op-
		posed to the regular {}N code, we do pop the
		binary number. This allows us to enter the
		code here with fixed sizes, as in ONE{}POLY
		and TWO{}POLY.
0A7006	^>POLY	( Meta $ ightarrow$ Q )
		Builds polynomial.
0A1006	^>TPOLY	( P ob $\rightarrow$ P' )
		Replaces >TCOMP for polynomial building.

Addr.	Name	Description
0A2006	^>HPOLY	( P ob $\rightarrow$ P' )
		Replaces > HCOMP for polynomial building.
0A3006	^>TPOLYN	( P ob1 obn $\#n \rightarrow P'$ )
		Improved >TCOMP for polynomial building.
0A4006	^>HPOLYN	( P ob1 obn $\#n \rightarrow P'$ )
		Improved >HCOMP for polynomial building.
0A5006	^MKPOLY	( $\#n \ \#k \rightarrow P$ )
		Makes polynomial of nth variable to the power
		k.
2AB006	^MAKEPROFOND	$( ob \# \rightarrow \{\{\{\ldots\{o\}\ldots\}\}\} )$
		Embedds ob in the given number of lists.
4F4006	^TRIMext	$(Q \rightarrow Q')$
		Removes unnecessary zeros from polynomial.
4F5006	^PTrim	$( ob \rightarrow ob' )$
		Trims polynomial.
0A6006	^ONE>POLY	$(Q \rightarrow Q')$
		Increases variable depth. Constants (Z,Irr,C)
	_	are not modified.
302006	^TCHEBext	$(zint \rightarrow P)$
		Tchebycheff polynomial. If zint>0 then 1st
2==226	^	kind, if <0 then second kind.
3DE006	^LRDMext	$(P \# \rightarrow [])$
		Left ReDiMension. Adds 0 to the left of polyno-
255006	^	mial to get a symbolic vector of lenght #+1.
3DF006	^RRDMext	$\{\} \ \# \to \{\}\ $
		Right ReDiMension: like LRDM but 0 at the
20000	^DEGDE	right and {}.
3E0006	^DEGREext	( {} → degre )
211006	^EHODNED	Degree of a list-polynomial. ( $P/d r \rightarrow P[X]_{div_[X-r]/d r P[r]/d$
3E1006	^FHORNER	$(P/QI \rightarrow P[X]_QIV_[X-I]/QI P[I]/Q$
		Horner scheme.
3E2006	^HORNext	( $P r \rightarrow P[X]_{div}[X-r] r P[r]$ )
252000	HOMMEYC	Horner scheme.
3E4006	^MHORNext	( $P r \rightarrow P[X]_{div}[X-r] r P[r]$ )
254000	MITOTALENC	Horner scheme for matrices.
		HOLLIGI SCHOLLE TOL HEAVETCES.

Addr.	Name	Description
3E6006	^LAGRANGEext	( M $ ightarrow$ symb )
		Lagrange interpolation. Format of the matrix
		is
		[ [x1 xn] [f(x1) f(xn)]
	_	Returns a polynomial P such that P(xi)=f(xi)
10F007	^RESULTANT	$(P1 P2 \rightarrow P)$
		Resultant of two polynomials. Depth of P is one
110000	^	less than depth of P1 and P2.
110007	^RESULTANTLP	( res g h P1 P2 $\rightarrow$ +/-res g' h' P1'
		P2')
111007	^DEGDGIITEMO	Subresultant algorithm innerloop. ( $P O \rightarrow P'$ )
111007	^RESPSHIFTQ	$(PQ \rightarrow P')$ Resultant of P and Q shifted. $gcd[Q(x-$
		r),P(x)]!=1 equivalent to r root of P' P' has same
		depth than P and Q.
112007	^ADDONEVAR	( $P \rightarrow P'$ )
112007	ADDONEVAR	Adds one variable just below the main var.
		works for polynomial, not for fractions.
0CF007	^SHRINKEVEN	( $P \rightarrow P'$ )
001 007	SHICHWICH V HIV	Changes var Y=X^2 in an even polynomial.
0D1007	^SHRINK2SYM	$( N D \rightarrow N' D' )$
		Shrinks 2 polynomials using symmetry proper-
		ties.
0D2007	^SHRINKSYM	$(N \rightarrow N')$
		Shrinks 1 polynomial using symmetry proper-
		ties. Degree of N must be even. If it is odd then
		N should be divided by X+1.
0D3007	^SHRINK2ASYM	( N D $\rightarrow$ N' D' )
		Shrinks 2 polynomials using antisymmetry
		properties.
0D4007	^SHRINKASYM	$(N \rightarrow N')$
		Shrinks 1 polynomial using antisymmetry
		properties. Degree of N must be even. If it is
		odd then N should be divided by X+1.
103006	^PNMax	$(P \rightarrow Z)$
161006	^	Gets the coefficient of P with max norm.
161006	^SWAPNDXF	( Qden Qnom → symb )
		Builds a symbolic from rational polynomial.

Addr.	Name	Description
162006	^NDXFext	( Qnom Qden $ ightarrow$ symb )
		Builds a symbolic from rational polynomial.
163006	^SWAPFXND	( symb ob $ ightarrow$ ob Qnom Qden )
		Converts symbolic to rational polynomial.
164006	^FXNDext	( $symb   o  Qnom  Qden$ )
		Converts symbolic to rational polynomial.
3D7006	^REGCDext	( a b $\rightarrow$ d u v au+bv=d )
3D8006	^EGCDext	( a b $\rightarrow$ d u v au+bv=d )
		Bezout identity for polynomials.
0EA006	^PEvalFast?	( Z Pn $ ightarrow$ Z Pn F / Pn[Z] T )
		Attempts to evaluate Pn at X1=Z using fast
		register arithmetic. Fails if any of the follow-
		ing is true: Pn is not sunivariate; Z is polyno-
		mial after all; Z size is too big for register; Any
		overflow occurs during Horner evaluation.
10E007	^FLAGRESULTANT	$($ symb1 symb2 $\rightarrow$ symb $)$
		Resultant of two polynomials in symbolic form.

### 46.1.4 Tests

Addr.	Name	Description
10B006	^Univar?	( P $\rightarrow$ P flag )
		Tests if polynomial is univariate.
10C006	^SUnivar?	( P $ ightarrow$ P flag )
		Tests if polynomial is univariate and the coeffi-
		cients are bounded by register size.
0CC007	^POLYPARITY	( poly $ ightarrow$ Z )
		Tests if a polynomial (internal rep) is
		even/odd/none. Z=1 if even, -1 if odd, 0 if
		neither even nor odd.
0D6007	^POLYSYM	( P $ ightarrow$ Z )
		Tests symmetry of coefficients of polynomial.
		Z=1 for symmetric, -1 for anti, 0 otherwise.
0D7007	^POLYASYM	( P $ ightarrow$ Z )
		Tests "antisymmetry" of coef of polynomial. Z=1
		for symmetric, -1 for anti, 0 otherwise.

# **Chapter 47 Root Finding**

In this chapter you will find entries related to finding roots of equations.

## 47.1 Reference

## 47.1.1 Root Finding and Numerical Solvers

Addr.	Name	Description
272006	^MULMULText	( {} % → {}')
		Multiplies multiplicities in a factor list by co-
		eff.
274006	^METAMM2	( meta % $ ightarrow$ meta' )
		Multiplies by % all multiplicities of meta.
275006	^COMPLISText	$( \ \{\} \ \rightarrow \ \{\}' \ )$
276006	^METACOMPRIM	( Meta $ ightarrow$ Meta' )
		Suppresses multiple occurrances of the same
		factor by adding corresponding multiplicities.
278006	^METACOMP1	( f1fk-1 mk-1 meta-res mk fk $\#$
		ightarrow f1fk-1 mk-1 meta-res )
279006	^ADDLISText	( $\{\}$ %n ob $\rightarrow$ $\{\}$ ')
		Adds ob with multiplicity %n to the list.
		Checks if ob is in {}.
27A006	^DIVISext	( ob $ ightarrow$ {divisors} )
		Returns list of divisors of ob.
27B006	^FACTlext	( symb-poly $ ightarrow$ Lvar Q $\{\}$ )
		{} is the list of root/multiplicity of sym with
		respect to the current variable.
27C006	^FACTOext	( $symb   o  Lvar   Q   \{\}$ )
		{} is the list of factors/multiplicity of symb.
27D006	^ZFACTO	( C $ ightarrow$ {} C Lfact )

Addr.	Name	Description
27E006	^SOLVext	( symb $\rightarrow$ {} )
		Numeric solver for univariate polynomials.
		The list contains the roots without multiplic-
		ity.
27F006	^FRND	$(ob \rightarrow ob'))$
		Float rounding for %%, C%% or list of either
		type. Used by SOLVext to reconstruct factors.
280006	^BICARREE?	( P $\#5 \rightarrow \text{meta cst\_coeff T}$ )
200000	DICARREE:	$(P #5 \rightarrow P #5 F)$
		$(P \# \rightarrow P \# F)$
		Searches if P is a bisquared 4-th order equa-
		tion. Returns a meta of factors and the mul-
		tiplying coeff in that case.
281006	^REALBICAR	( f1 #1 coef $ ightarrow$ meta rest T )
113007	^IROOTS	( P $ ightarrow$ list )
		Finds integer roots of a polynomial.
283006	^EVIDENText	( P $\rightarrow$ meta cst_coeff )
		Returns the roots of a polynomial P. Calls the
	<b>^</b>	numeric solver.
284006	^EVIDSOLV	( P → meta cst_coeff )
		Returns the roots of a 1st, 2nd order and some other poly. Calls the numeric solver if
		exact solving fails.
285006	^DEG2ext	exact solving rans. ( $P \rightarrow \{\}$ )
203000	DHOZCAC	Returns the roots of a 2nd order polynomial.
286006	^METADEG2	( $P \rightarrow P \text{ meta}$ )
		Returns the roots of a 2nd order polynomial.
		P must be of order 1 or 2.
287006	^METADEG1	( P $ ightarrow$ P meta )
		Returns the roots of a 1st order polynomial.
		P must be of order 1.
288006	^DEG1	$(f \rightarrow r)$
		Root of a first order factor. f is one level depth
00000	^= <u></u>	deeper than r.
289006	^FDEG2ext	(P → meta-fact cst_coef)
		Returns factors of a 2nd order polynomial and the corresponding multiplying coeffi-
		cient. tests for 1st order polynomial.
		cicii. icaia idi tai di dei poiyildililai.

Addr.	Name	Description
28B006	^RACTOFACext	$(r \rightarrow n d)$ Converts root to factor. Factor is n/d, one level depth deeper than r.
28C006	^FACTORACext	( f $\rightarrow$ r cst_coef ) Converts a factor to a root, solving 1st order factor. f and cst_coef are one level depth deeper than r.
28D006	^RFACText	( ob # → {} intob meta ) {} is the list of variables. Meta is made of roots or factors of numerator (N) or denomenator (D) or both (N/D), depending on #. ZERO for roots N/D; ONE for roots N; TWO for roots D with numeric solver call; THREE for roots D without num. solver call; FOUR for factors N/D; FIVE for factors N; SIX for factors D with numeric solver call; SEVEN for factors D without num.solver call.
28E006	^RFACT2ext	( ob $\{\} \# \to \{\}$ intob meta ) Like RFACText, but the list of variables is given.
28F006	^RFACTSTEP3	( ob → meta-fact ) Partial square-free factorization w.r.t. the main variable. Extract trivial factors Etape 3 ob → meta-fact.
290006	^RFACTSTEP5	( %m on $\rightarrow$ add-to-meta-res ) Factorization of a square-free polynomial.
291006	^METASOLV	( pn cst_coeff → meta cst_coeff ) Non-integer factorization (sqrt extensions and numeric). multiplicty is in LAM 5,.
293006	^METASOLV2	( cst_coeff p $\rightarrow$ fr1 %m [fr2 %m] # cst_coeff ) Returns roots/factors of 1st and 2nd order polynomials.
294006	^METASOLV4	( cst1 f1 fk $\#k$ cst2 $\rightarrow$ fr1 $\#k$ frn $\#k$ $\#k$ cst_coeff ) Returns factors or convert to roots if needed. #k=1,2 or 4, fk are of order 1 or 2.

Addr.	Name	Description
295006	^ADDMULTIPL	( meta cst_coeff $\rightarrow$ meta' cst_coeff
		Adds multiplicities to a meta. Multiplicity is
		in LAM 5.
296006	^FACTOOBJext	( { fact mult } flag prg* prg^ →
		ob)
		Rebuilds an object from its list of factors
		(flag=TRUE) or roots (flag=FALSE) using prg* to multiply and prg^ to take multiplicity
		power.
093006	^ALG48MSOLV	( Lp $\rightarrow$ Lidnt Lsol )
		Calculates Groebner basis multivar solution.
		LAM3 must be bound to Lvar and LAM4 to
	<b>2</b>	Lidnt.
094006	^GMSOLV	( Lp $\rightarrow$ meta-sol ) Calculates Groebner basis multivar solu-
		tions. LAM1 must be bound to the num-
		ber of vars A solution is a list { o1 on }
		where #n=LAM1 ok embedded in k-1 lists is
		the value of the k-th var ok may be undef.
095006	^GBASIS	$(Lp \rightarrow G)$
		Calculate Groebner basis.
		$G = \{ 1 \}$ if no solutions $G = \{ 0 \}$ if identically true.
096006	^GSOLVE	$G = \{ 0 \}$ if identically true. ( $Lp \rightarrow Lg$ )
0,0000	CDOLVI	Calculate factorized Groebner basis.
		Lg = { Lg1 Lg2 Lgn }
		Lgi = independent solution
		(probably)
		Lg = {} if no solutions
		Lg = { { 0 } } if identically
097006	^GFACTOR	true. ( Lp fctr? $\rightarrow$ Lq )
	<del>-</del> -	Calculate Groebner basis or factorized Groeb-
		ner basis. Redundant bases are not removed.
099006	^REDUCE	( p G $\rightarrow$ q )
		Reduces polynomial with respect to given ba-
		sis.

Addr.	Name	Description
09A006	^FASTREDUCE	$( r P \rightarrow q T / r P F )$
		Assembly version of REDUCE for polynomials
		with short coefficients. Returns FALSE if an
		overflow occurs during the reduction. As-
		sumes r is a genuine polynomial (not con-
		stant). Assumes G is not empty. Assumes G
		does not contain zeros (is trimmed).
37D006	^ROOTM2ROOT	$( \{ \}/V \rightarrow V' )$
		Transforms list of root/multiplicites to vector
		of roots.
0F2007	^PASCAL_NEXTLINE	$( \ \{\} \ \rightarrow \ \{\} \ ' \ )$
		Finds next line in the Pascal triangle.
0F3007	^DELTAPSOLVE	$(Q \rightarrow P)$
		Solves $P(x+1)-P(x)=Q(x)$ . Internal polynomial
		function.

# **Chapter 48 Calculus Operations**

The entries in this chapter are related to several aspects of Calculus, such as limits, derivates, partial fraction expansions and Laplace transformations.

### 48.1 Reference

### 48.1.1 Limits and Series Expansion

Addr.	Name	Description
46F006	^SYMTAYLOR	( symb id $%/z \rightarrow symb$ )
		Taylor series expansion around point 0
		(McLaurin's series) with regard to given vari-
		able, and of the given order.
471006	^TRUNCDL	( DL-l reste-l $ ightarrow$ truncated_DL )
		Series expansion truncation.
472006	^LIMSERIES!	( expression X=a $ X$ % $ zint  o )$
		a lim DL-l rest-l num-l/deno-l equiv-l lvar #
		Series expansion. #=1 for X=a-h or X=-1/h.
477006	^LIMIT!	( symb $ ightarrow$ DL-1 reste-1 num-1/deno-1
		equivl lim. lvar flag )
		<pre>lim. = { symf direction }</pre>
478006	^LIMSTEP1!	( symb $ ightarrow$ { DL-l reste-l
		<pre>num-l/deno-l equivl } flag )</pre>
47C006	^LIMLIM!	( $\#$ lvar equiv-l $ ightarrow$ lvar lim )
47F006	^LIMCMPL!	( reste-1-l reste-2-l $ ightarrow$ reste-l )
480006	^LIMEQUFR!	( $n/d \# \rightarrow n/d-l \ equiv \%$ )
481006	^LIMEQU!	( $\{\}$ # $\rightarrow$ $\{\}$ / $\{\}$ -equiv-l $\{\}$ -equiv-l
		{ # # # } )

Addr.	Name	Description
483006	^LIM+-!	( DL1DLn $\#$ n op $ o$ DL flag )
		DL = { DL-l reste-l num-l/deno-l equiv-l }.
48C006	^LIMDIVPC!	( $\#$ ordre num-l deno-l $\to$ num-l
		deno-l )
48E006	^LIMPROFEND!	( num deno $\#prof \rightarrow num deno$ )
490006	^LIM%#!	( num-l deno-l $\{\$\$\}  ightarrow$ num-l'
		deno-l' #prof {%%} )
49E006	^LIM#VARX!	( lvar lvar $ ightarrow$ #varx )
4A1006	^HORNEXP!	( lim lvar X-l reste-l $ ightarrow$ lvar DL
		reste-l )
4B6006	^VARCOMP!	( var1 var2 $ ightarrow$ flag )
4BA006	^VARCOMP32!	( var $\rightarrow$ 0: )
4BD006	^LIMVALOBJ!	( ob lvar $ ightarrow$ symb )
4BE006	^LIMVAL!	( ob $ ightarrow$ coeff val )
4BF006	^EQUIV!	( $\{\}$ lequiv $ ightarrow$ equiv ordre )
4C0006	^LVARXNX2!	( ob $ ightarrow$ ob lvarx lvarnx )
4C2006	^FindCurVar	( $symb \rightarrow symb$ )
		Sets a new current var if needed.
4C3006	^LIMVAR!	( $symb   o  symb$ lvar )
15C006	^RISCH13	$( \{\}/\{\}' \rightarrow \{\}'')$
		Assuming {}' has length 1, divides all elements
		of {} by this element. Used by RISCHext and
		by SERIES to have a nicer output of series.
49E006 4A1006 4B6006 4BA006 4BD006 4BE006 4C0006 4C2006	^LIM#VARX! ^HORNEXP!  ^VARCOMP!  ^VARCOMP32! ^LIMVALOBJ! ^LIMVAL! ^EQUIV! ^LVARXNX2! ^FindCurVar  ^LIMVAR!	<pre>deno-l' #prof {%%}) ( lvar lvar → #varx ) ( lim lvar X-l reste-l → lvar DL reste-l ) ( var1 var2 → flag ) ( var → 0: ) ( ob lvar → symb ) ( ob → coeff val ) ( {} lequiv → equiv ordre ) ( ob → ob lvarx lvarnx ) ( symb → symb ) Sets a new current var if needed. ( symb → symb lvar ) ( {}/{}' → {}'' ) Assuming {}' has length 1, divides all elements of {} by this element. Used by RISCHext and</pre>

### 48.1.2 Derivatives

Addr.	Name	Description
3DC006	^PDer	$(\ \{\}\  o\  exttt{der}\ )$
1A1006	^DERIVext	( ob id $ ightarrow$ ob' )
		( ob sym $ ightarrow$ ob' )
		( ob V $ ightarrow$ V' )
		Calculates the derivative of the object. For
		a list argument calculates the gradient with
		respect to the variables in the list. If the vari-
		able is a symbolic, the first variable in it is
		used. Note that the gradient is a vector quan-
		tity, thus the result is returned as a list.

Addr.	Name	Description
1A3006	^DERIVIDNT	( ob id $\rightarrow$ ob' )
		Main entry point for derivative with respect
		to a identifier.
1A4006	^DERIVIDNT1	$( ob \rightarrow ob' )$
		Main entry point for derivative with respect
		to the identifier stored in LAM1.
1A5006	^DERIV	$( symb \rightarrow symb' )$
		Derivative of symb with respect to the vari-
		able stored in LAM1.
1A6006	^METADERIV	( Meta $ ightarrow$ Meta' )
		Derivative of Meta object.
1BD006	^METADER&NEG	( Meta $ ightarrow$ Meta' )
		Meta derivative and negate.
1A9006	^METADER+	( Meta&+ $ ightarrow$ Meta' )
		Meta derivative of addition.
1AA006	^METADER-	( Meta&- $ ightarrow$ Meta' )
		Meta derivative of subtraction.
1AB006	^METADER*	( Meta&* → Meta' )
		Meta derivative of multiplication.
1AC006	^METADER/	( Meta&/ → Meta' )
		Meta derivative of division.
1AD006	^METADER^	( Meta&^ → Meta' )
	<b>^</b>	Meta derivative of power.
1AE006	^METADERFCN	( Meta → Meta' )
1006	^	Meta derivative of a function.
1AF006	^METADERDER	( symb_id_; sym_fcn_; xDER #3 →
		Meta')
150000	^14==1 0=0 = 4	Meta derivative of a derivative of a function.
1B0006	^METADERI4	( Meta → Meta' )
1B1006	^METADERI3	<pre>Meta derivative of a defined integral. ( Meta → Meta' )</pre>
IBIUUO	METADERI3	Meta → M
1B2006	^METADERIFTE	( Meta → Meta' )
102000	METADEKTITE	Meta derivative of IFTE.
1B4006	^METADEREXP	( Meta → Meta' )
10100		Meta derivative of EXP.
1B5006	^METADERLN	( Meta → Meta' )
	·	Meta derivative of LN.
		2.2000 0022700770 01 2271

Addr.	Name	Description
1B6006	^METADERLNP1	( Meta → Meta' )
		Meta derivative of LNP1.
1B7006	^METADERLOG	( Meta $ ightarrow$ Meta' )
		Meta derivative of LOG.
1B8006	^METADERALOG	( Meta $ ightarrow$ Meta' )
		Meta derivative of ALOG.
1B9006	^METADERABS	( Meta $ ightarrow$ Meta' )
		Meta derivative of ABS.
1BA006	^METADERINV	( Meta $ ightarrow$ Meta' )
		Meta derivative of INV.
1BB006	^METADERNEG	( Meta $ ightarrow$ Meta' )
		Meta derivative of NEG.
1BC006	^METADERSQRT	( Meta $ ightarrow$ Meta' )
		Meta derivative of SQRT.
1BE006	^METADERSQ	( Meta $ ightarrow$ Meta' )
		Meta derivative of SQ.
1BF006	^METADERSIN	( Meta $ ightarrow$ Meta' )
		Meta derivative of SIN.
1C0006	^METADERCOS	( Meta $ ightarrow$ Meta' )
		Meta derivative of COS.
1C1006	^METADERTAN	( Meta $ ightarrow$ Meta' )
		Meta derivative of TAN.
1C2006	^METADERSINH	( Meta $ ightarrow$ Meta' )
		Meta derivative of SINH.
1C3006	^METADERCOSH	( Meta $ ightarrow$ Meta' )
		Meta derivative of COSH.
1C4006	^METADERTANH	( Meta $ ightarrow$ Meta' )
		Meta derivative of TANH.
1C5006	^METADERASIN	( Meta $ ightarrow$ Meta' )
		Meta derivative of ASIN.
1C6006	^METADERACOS	( Meta $ ightarrow$ Meta' )
		Meta derivative of ACOS.
1C7006	^METADERATAN	( Meta $ ightarrow$ Meta' )
		Meta derivative of ATAN.
1C8006	^METADERASH	( Meta $ ightarrow$ Meta' )
		Meta derivative of ASINH.
1C9006	^METADERACH	( Meta $ ightarrow$ Meta' )
		Meta derivative of ACOSH.

Addr.	Name	Description
1CA006	^METADERATH	( Meta → Meta' )
		Meta derivative of ATANH.
1B3006	^DERARG	( meta-symb $ ightarrow$ argl argk derl
		derk #k op )
		Finds derivative of arguments.
1CB006	^pshder*	( Metal Meta2 $\rightarrow$ Meta2&Meta1'&* )
		Meta derivative utility.
1CC006	^SQRTINVpshd*	( Metal Meta2 $ ightarrow$
		<pre>Meta2&amp;SQRT&amp;INV&amp;Meta1'&amp;* )</pre>
		Meta derivative utility.

## 48.1.3 Integration

Addr.	Name	Description
07F007	^ODE_INT	( symb idnt $ o$ symb )
		Integration with addition of a constant.
2C5006	^IBP	$(u'*vu \rightarrow u*v-u*v')$
		Internal integration by parts. If u is a constant
		return INTVX(u'*v)+u. If stack 2 is a list it must
		be of the form { olduv u'*v } then olduv will
		be added to u*v at stack level 2. This permits
		multiple IBP in algebraic mode, e.g.
		$IBP(ASIN(X)^2, X)$
		<pre>IBP(ANS(1),sqrt(1-X^2))</pre>
		IBP(ANS(1),C) the last step with an
		integral
		containing a cst C.
2D0006	^PREVALext	( symb inf sup x $\rightarrow$ symb $ $ x=sup -
		symb x=inf)
		Evaluates an antiderivative between 2 bounds
		Does not check for discontinuities of symb in this
		interval.
2D1006	^WARNSING	( symb inf sup $vx \rightarrow symb$ inf sup $vx$ )
		Warns user for singularity.
2D2006	^INText	$( symb x \rightarrow int[\$,x, symb, xt] )$
		Return unevaluated integral.

Addr.	Name	Description
2D3006	^INT3	( $f(x) \times y \rightarrow F(y)$ where $F'=f$ )
		Undefined integration. No limit for underdetermined form.
3DD006	^INTEGRext	( $\{\}  ightarrow  exttt{prim}$ )

## **48.1.4 Partial Fractions**

Addr.	Name	Description
3D2006	^PARTFRAC	( o $ ightarrow$ symb )
		Partial fraction expansion of o with respect to
		the current variable.
3D3006	^INPARTFRAC	( o list $ o$ symb )
		Partial fraction expansion of o. lvar must be
		bound to LAM2, list is =lvar if o is in external
		format. list is NULL{} if o is still in internal for-
		mat.

# 48.1.5 Differential Equations

Addr.	Name	Description
07E007	^DESOLVE	( list symb1 $\rightarrow$ list_sols )
		( symb symb1 $ ightarrow$ list_sols )
		Solves ordinary differential equation. For some
		ode's returned level2 is not symb1.
081007	^LDECSOLV	( $2nd\_member char\_eq \rightarrow solution$ )
		Linear differential equation with constant coef-
		ficients.
082007	^LDEGENE	( eq. carac $ ightarrow$ sol generale )
083007	^LDEPART	( 2nd membre, eq carac $ ightarrow$ eq. carac,
		sol part )
084007	^LDSSOLVext	( $V M \rightarrow V'$ )
		M is the matrix of the system. V is the vector of
		the 2nd members.
085007	^ODETYPESTO	( type $ ightarrow$ )
		Store ode type in variable ODETYPE.

Addr.	Name	Description
086007	^ODE_SEPAR	( $symb \to symb  symb-y  symb-x  T$ )
		( $symb \;  o \; symb \; F$ )
		Tries to separate symb as a product of a function
		of y and a function of x.

# 48.1.6 Laplace Transformation

Addr.	Name	Description
087007	^LAPext	( $symb \rightarrow symb'$ )
		Laplace transform for polynomial*exp/sin/cos. Re-
		turns LAP() for unknown transforms.
088007	^ILAPext	(  symb   o  symb' )
		Inverse Laplace transform for rational fractions.
		Delta functions for the integral part.
08B007	^ILAPEXP	( ck rk $\rightarrow$ ck*exp[rk*x] )

# **Chapter 49 Summation**

In this chapter, you will find the main entries related to summation, and also some sub-routines used by those entries.

Addr.	Name	Description
0F9007	^SUM	( $sym \; idnt \;  o \; sym$ )
		Internal SUM. The variable can be specified.
0FB007	^SUMVX	( $\text{sym} \rightarrow \text{sym}$ )
		Internal SUMVX. Works always with respect to
		the current variable.
0FD007	^RATSUM	$(sym \rightarrow sym)$
		Discrete rational sum.
0FE007	^FTAYL	( f shift $\rightarrow$ f' )
		Taylor shift for rational fractions.
0FF007	^CSTFRACTION?	( ob $\rightarrow$ ob flag )
		Taylor shift for rational fractions. Returns
		TRUE if ob is a cst fraction.
104007	^HYPERGEO	$( \text{symb} \rightarrow \text{symb})$
		Tests and does hypergeometric summation.
100007	^NONRATSUM	$(z/symb \rightarrow symb)$
		Discrete summation (hypergeometric case).
103007	^meta_cst?	( meta $ ightarrow$ meta flag )
		Tests for meta to be cst with respect to cur-
		rent var.
108007	^ZEILBERGER	$(f(n,k) n k d \rightarrow C T)$
		$(f(n,k) n k d \rightarrow F)$
		Zeilberger algorithm * NOT IMPLE-
		MENTED YET*.

394 49. Summation

Addr.	Name	Description
109007	^SYMPSI	( $sym \rightarrow Psi(x)$ )
		Digamma function.
10B007	^SYMPSIN	( sym int $\rightarrow$ Psi(x,n) )
		Digamma function.
11C007	^%%PSI	( %%x -> %% )
		Digamma function.
10D007	^IBERNOULLI	( $\#/zint \rightarrow Q$ )
		Bernoulli numbers.
0D9007	^NDEvalN/D	( num deno n d $ ightarrow$ num' deno' )
		Evals list poly over a list fraction.
0DA007	^PEvalN/D	( P n d $\rightarrow$ num d $\#$ )
		Evals list poly over a list fraction.
3C1006	^vgerxssSYMSUM	( Meta2 Meta1 $ ightarrow$ meta )
		Symbolic sum with tests for two zints.
		lam'sumvar bound to 'id/lam' and
		lam'sumexpr to 'expr'.

# **Chapter 50 Modular Operations**

The entries in this chapter are related to modular arithmetic and other modular operations.

## 50.1 Reference

## 50.1.1 Modulo Operations

Addr.	Name	Description
252006	^FLAGFACTORMOD	( $symb \to symb$ )
		FACTOR modulo.
253006	^MFACTORMOD	( $M \rightarrow M'$ )
		FACTOR modulo for amtrices.
256006	^LIFCext	( $\{ ext{contfrac}\}  ightarrow  ext{fraction}$ )
		Converts continued fraction to rational.
0E1006	^PEvalMod	( Q Z Zn $\rightarrow$ Q' )
		Computes value of polynomial mod Zn.
0E2006	^QAddMod	( Q1 Q2 $Zn \rightarrow Q'$ )
		Polynomial addition modulo Zn.
0E3006	^QSubMod	( Q1 Q2 $Zn \rightarrow Q'$ )
		Polynomial subtraction modulo Zn.
0E4006	^QMulMod	( Q1 Q2 $Zn \rightarrow Q'$ )
		Polynomial multiplication modulo Zn.
0E5006	^QDivMod	( Q1 Q2 Zn $ ightarrow$ Qquo Qrem )
		Polynomial division modulo Zn. In regular di-
		vision the coefficients in the remainder can
		increase very quickly to tens of digits, thus
		it is important to normalize the coefficients
		whenever possible.

Addr.	Name	Description
0E6006	^QInvMod	( Q $Zn \rightarrow Q'$ )
		Polynomial inversion modulo Zn.
0E7006	^QGcdMod	( Q1 Q2 Zn $\rightarrow$ Q' )
		Polynomial GCD modulo Zn for univariate
		polynomials. The result is made monic.
4C5006	^ISOL1	( symb id $ ightarrow$ id symb' )
4C6006	^ISOLALL	( symb id $ o$ id $\{\}$ )
		Internal SOLVE.
4C7006	^ISOL2ext	( symb id $ o$ symb' )
		( symb id $ ightarrow$ {} )
		Like ISOL1 if isolflag is set. Otherwise re-
		turns the list of all found solutions.
4C8006	^BEZOUTMSOLV	( Lpoly Lidnt $ ightarrow$ Lidnt sols )
		If no extension in Lpoly, calls ALG48 GSOLVE
		Otherwise, solves by Bezout "Gaussian" elim-
		ination. In the latter case, if system seems
		underdetermined, Lidnt is truncated. Then
		the system must be exactly determined and
		polynomials must be prime together.
4C9006	^ROOT{}N	( meta of roots $ ightarrow$ list of roots )
		Drops tagged roots.
4CA006	^MHORNER	( poly-l $\{r1rk\} \# \rightarrow P[r1rk]$ )
		Top-level call. Poly-l might be a matrix.
4CB006	^MHORNER1	( P $\{ r \} \rightarrow P[r]$ )
4CC006	^SQFFext	( Q $ ightarrow$ { F1 mult1 $\dots$ Fn multn $\}$ )
4CD006	^MSQFF	( Q $ ightarrow$ F1 mult1 Fn multn $\#2n$ )
		Full square-free factorization of object. The
		result is given as a Meta object.
4CE006	^%1TWO	( ob $ ightarrow$ ob %1 #2 )
		Square free factorization of unknown (?) ob-
		ject. See MSQFF.
4CF006	^MZSQFF	( Z $ ightarrow$ Z1 mult1 Zn multn #2n )
		Full factorization of an integer.

Addr.	Name	Description
4D0006	^MZSQFF1	( Meta curfac %n newfac T $\rightarrow$ Meta curfac %n+1 ) ( Meta curfac %n newfac F $\rightarrow$ Meta' newfac %1 ) Adds integer factor to factor list. If the
4D2006	^MLISTSQFF	factor is the same as the last time, only the multiplicity is increased.  ( $P \rightarrow Meta$ )
402000	пштэтэўгг	Full square-free factorization of a polynomial with a recursive call on the GCD of all coefficients.
4D3006	^METASQFFext	( P-list $\rightarrow$ S1 %1Se-1 %e-1 %e ee Te Re ) Square-free factorization.
4DE006	^LIDNText	( ob $\rightarrow$ {} ) Gets list of all ids present in ob.
4DF006	^LVARXNXext	( symb → symb x lvarnx lvarx ) Finds variable of symb depending on current variable and other variable. Using LVAR is impossible here because of sqrt.
4E0006	^ISPOLYNOMIAL?	<ul> <li>( ob → flag )</li> <li>Returns TRUE if symb is polynomial with respect to current variable.</li> </ul>
4E1006	^2POLYNOMIAL?	( symb1 symb2 → symb1 symb2 flag ) Returns TRUE if symb1 and symb2 are polynomial with respect to current variable.
4E2006	^VXINDEP?	( symb $\rightarrow$ symb flag ) Returns TRUE if symb is independent of current variable.
4E4006	^RLVARext	( ob $\rightarrow$ {} ) Recursive search of all variables.
4E5006	^LLVARDext	( o $ ightarrow$ #depth o lvar )
4E6006	^VXLVARext	( $symb   o  symb$ lvar )
4E7006	^LVARext	( ob $\rightarrow$ ob {} ) List of variables. Square roots <i>are</i> included in the list of rational operators.

Addr.	Name	Description
4E8006	^VX>LVARext	( ob → ob {})
		Like LVARext but the current variable is
		added using >HCOMP. Square roots are in-
	A	cluded in the list of rational operators.
4E9006	^VX>	$(\ \{\}\ \rightarrow\ \{\}\ )$
		If VX is in the list then moves it to the begin-
453006	^+	ning of the list. Otherwise does nothing.
4EA006	^VX!	$(\ \{\}\  o\ \{\}\ )$
		If VX is in the list then moves it at the begin-
		ning. Otherwise VX is added to the beginning of the list.
45000	^	
4EC006	^LIDNTLVAR	( symb lidnt → symb lidnt lvar )
		lvar is the list of variables in symb, but ele-
		ments of lidnt are moved to the beginning of lvar.
4ED006	^LISTOPRAC	$( \rightarrow \{ \} )$
450000	LISTOPRAC	Returns the list of rational operator with sqrt
		appended to the list.
4EE006	^LISTOPext	$( \rightarrow \{ \} )$
TEECOO	HISTOPEAC	List of basic "rational" operators without
		square root.
4EF006	^LISTOPSQRT	$( \rightarrow \{\} )$
111 000	HIDIOI DQIKI	List of basic "rational" operators with square
		root.
4F0006	^LVARDext	( ob listop → lidnt )
		( Meta listop → lidnt )
		Determines list of variables in ob (or Meta)
		using the given list of basic "rational" opera-
		tors.
4F2006	^DEPTHext	( ob $\rightarrow$ $\#$ )
		Returns the max number of embedded lists in
		ob.
4F3006	^DEPTHOBJext	( objet $\#  o  ext{depth}$ )
4F6006	^TRIMOBJext	( ob $ ightarrow$ ob ' )
		Trims object.

Addr.	Name	Description
4F7006	^NEWTRIMext	( $Q \rightarrow Q$ )
		Recursively tests if Q is a list of one con-
		stant element. This is much faster than
		TRIMOBJext and sufficient for the output of
		programs which are trimmed on the fly.
4F8006	^>POLYTRIM	( $meta \rightarrow \{\}$ )
		Equivalent to { } POLY TRIMOBJext.
4F9006	^ELMGext	$( ob \rightarrow ob' )$
		Trims small numbers (less than epsilon).
0E9006	^IsV>V?	( v1 v2 $ ightarrow$ flag )
		Returns TRUE if v1 is lexicographically after
		v2.
0EB006	^PZadic	$(QZ \rightarrow Q')$
104006	^LISTMAXext	( P $ ightarrow$ P Z T depth )
		( P $\rightarrow$ P ? F $\sharp 0$ )
		Step 1 for gcdheu: Returns FALSE if gcd-
		heu can not be applied (e.g. if P contains
		irrquads). Returns TRUE otherwise, Z is the
		max of all integers of P or 2*max if there are
		complex in P.
0EC006	^GCDHEUext	( A B $\rightarrow$ a b c pr[pgcd] A'/G' B'/G'
		flag )
		Heuristic GCD.

# **Chapter 51 Sign Tables**

A sign table is a list which describes thes sign of a expression in different intervals of a parameter. The list has an odd number of elements and looks like this:

```
{ value1 sign1.2 value2 sign2.3 ...signN-1.N valueN }
```

The values are key values of the parameter, usually  $-\infty$ ,  $+\infty$ , and the locations of singularities or zeros in the expression. The values must be ordered and can be numbers or symbolic expressions. The signs show the sign of the expression in the interval between the adjacent values. Signs are '-', '+', and '?' (if the sign is unknown). To compute the sign table of an expression with respect to the current variable, use the entry SIGNElext. For example, the sign table of the expression ' $X^2-1$ ' is

```
\{ -\infty '+' -1 '-' 1 '+' +\infty \}
```

Below is a list of the entries related to sign tables.

Addr.	Name	Description
237006	^SIGNE	( $symb \rightarrow sign$ )
		Compute the sign table of the expression with
		respect to the current variable. Internal ver-
		sion of the UserRPL command SIGNTAB.
0DC007	^SIGNE1ext	( expr $ ightarrow$ sign $)$
		Sign table of a polynomial or rational expres-
		sion.
0DE007	^SIGNUNDEF	$(  ightarrow  ext{sign})$
		Returns undefined sign table.
0DF007	^SIGNPLUS	( $ ightarrow$ sign )
		Returns always positive sign table.

Addr.	Name	Description
0E0007	^SIGNMOINS	( $ ightarrow$ sign )
		Returns always negative sign table.
0E1007	^SIGNELN	( $sign \rightarrow sign$ )
		Returns ln of a sign table.
0E2007	^SIGNEEXP	$($ sign $\rightarrow$ sign' $)$
		Returns exp of a sign table.
0E3007	^SIGNESIN	$($ sign $\rightarrow$ sign' $)$
		Returns sin of a sign table.
0E4007	^SIGNECOS	$($ sign $\rightarrow$ sign' $)$
		Returns cos of a sign table.
0E5007	^SIGNETAN	( $sign \rightarrow sign'$ )
		Returns tan of a sign table.
0E6007	^SIGNEATAN	$($ sign $\rightarrow$ sign' $)$
		Returns atan of a sign table.
0E7007	^SIGNESQRT	$($ sign $\rightarrow$ sign' $)$
		Returns sqrt of a sign table.
0E8007	^SUBSIGNE	( sign min max $ o$ sign' )
		Truncates a sign table.
0E9007	^SIGNERIGHT	( sign ob $ ightarrow$ sign' )
		Places ob at the end of a sign table.
0EA007	^SIGNELEFT	( sign ob → sign' )
		Places ob at the beginning of a sign table.
0EB007	^>SIGNE	( sign → sign' )
		Prepends { -infinity? } to a sign table.
0EC007	^SIGNE>	( sign → sign' )
		Appends { ? +infinity } to a sign table.
0ED007	^SIGNMULText	( sign1 sign2 $\rightarrow$ sign' )
		Multiplies two sign tables.
0DB007	^POSITIFext	( ob $ ightarrow$ ob flag )
		Tries to determine if ob is positive. In internal
		representation, this depends on increaseflag so
		that x-1 is positive if increaseflag is cleared,
		negative otherwise, because x is assumed to
		tend to +infinity or zero.
0EE007	^ZSIGNECK	( ob $ ightarrow$ ob flag )
		Returns sign of an expression. Error if unable
		to find sign.

402 51. Sign Tables

Addr.	Name	Description
0F0007	^ZSIGNE	( ob $ o$ zint )
		Returns sign of an expression. zint=1 for +, -1
		for -, 0 for undef. Expression does not need to
		be polynomial/rational.
0F1007	^zsigne	( meta $ ightarrow$ zint )
		Returns sign of a meta symbolic. zint=1 for +,
		-1 for -, 0 for undef. Expression does not need
		to be polynomial/rational.
07D007	^CHECKSING	( symb inf sup vx $ ightarrow$ symb inf sup vx
		flag )
		Checks for singularities in expr.

# **Chapter 52 Errors**

The CAS error messages all have numbers starting with DE. You can get a full list in Appendix E.

Entries ^ERABLEERROR and ^GETERABLEMSG add DE00 to the message number, so you only specify the last two digits of the message number. You can naturally use the error commands described in Chapter 22 with the CAS errors, using the full error numbers.

Addr.	Name	Description
57E006	^ERABLEERROR	( ♯ → )
		Calls CAS Error.
57D006	^GETERABLEMSG	( # → \$ )
		Get string in erable messages table.
090006	^ErrInfRes	Error 305h
		Generates "Infinite Result" error.
091006	^ErrUndefRes	Error 304h
		Generates "Undefined Result" error.
092006	^ErrBadDim	Error 501h
		Generates "Invalid Dimension" error.
57F006	^CANTFACTOR	Error DE1Ch
		Generates "Unable to find factor" error.
580006	^TRANSCERROR	Error DE20h
		Generates "Not reducible to a rational ex-
		pression" error.
581006	^NONUNARYERR	Error DE21h
		Generates "Non unary operator" error.
582006	^INTERNALERR	Error DE26h
		Generates "CAS internal error" error.

404 52. Errors

Addr.	Name	Description
583006	^INVALIDOP	Error DE28h
		Generates "Operator not implemented (SE-
		RIES)" error.
584006	^ISOLERR	Error DE2Ah
		Generates "No solution found" error.
585006	^NONINTERR	Error DE2Ch
		Generates "No solution in ring" error.
586006	^INTVARERR	Error DE32h
		Generates "No name in expression" error.
587006	^Z>#ERR	Error DE35h
		Generates "Integer too large" error.
0EF007	^SIGNEERROR	Error DE36h
		Generates "Unable to find sign" error.
588006	^Z<0ERR	Error DE46h
		Generates "Negative integer" error.
589006	^VXINDEPERR	Error DE47h
		Generates "Parameter is cur. var. depen-
		dent" error.
58A006	^NONPOLYSYST	Error DE49h
		Generates "Non polynomial systrem" error.
58B006	^COMPLEXERR	Error DE4Dh
		Generates "Complex number not allowed"
		error.
58C006	^VALMUSTBE0	Error DE4Eh
		Generates "Polyn. valuation must be 0" er-
		ror.
58D006	^SWITCHNOTALLOWED	Error DE4Fh
		Generates "Mode switch not allowed here"
		error.
119007	^NONALGERR	Error DE50h
		Generates "Non algebraic in expression" er-
		ror.
58E006	^ERR\$EVALext	( seco $ ightarrow$ action )
58F006	^Sys1IT	( ob $\rightarrow$ )
		Execute object if display flag is set.

# **Chapter 53 CAS Configuration**

The entries in this chapter provide ways to configure the CAS operations. The configurations that can be done here are the same that can be done by the user via flags or the MODES input form.

Addr.	Name	Description
08F007	^CFGDISPLAY	$(\  ightarrow\ )$
		Display current configuration of the CAS.
090007	^NEWVX	$( \rightarrow )$
		Input new current variable from the user.
091007	^NEWMODULO	$( \rightarrow )$
		Input new modulo from the user.
092007	^SWITCHON	( $\#$ flag $ ightarrow$ )
		Asks the user if a certain mode may be
		switched on by toggling system flag #flag. Er-
		rors if the user does not want to switch.
093007	^SWITCHOFF	( $\#$ flag $ ightarrow$ )
		Asks the user is a certain mode may be
		switched off by toggling system flag #flag. Er-
		ror if the user does not want to switch.
094007	^FLAGNAME	$( \ \# \rightarrow \ \# \ \$ \ )$
		Find the name of a flag.
1DC007	(^PUSHFLAGS)	$(\  ightarrow\ )$
		Internal version of User PUSH command:
		stores the current flag settings and path in
		the CASDIR/ENVSTK variable.

Addr.	Name	Description
1DD007	(^POPFLAGS)	$( \rightarrow )$
		Internal version of User POP command: pops
		the last pushed flag settings and path from
		the CASDIR/ENVSTK variable.
095007	^COMPLEXON	$( \rightarrow )$
		Turns complex mode on. Depending on sys-
		tem flag 120, the user is asked first.
096007	^COMPLEXOFF	$( \rightarrow )$
		Turns complex mode off. Depending on sys-
		tem flag 120, the user is asked first.
097007	^EXACTON	$( \rightarrow )$
		Turns exact mode on. Depending on system
		flag 120, the user is asked first.
098007	^EXACTOFF	$( \rightarrow )$
		Turns exact mode off. Depending on system
		flag 120, the user is asked first.
099007	^COMPLEXMODE	$( \rightarrow )$
		Set complex mode, refresh configuration dis-
00-00-	^	play.
09A007	^SETCOMPLEX	$(\rightarrow)$
000000	^ GOMBI EIKO	Set complex mode.
09B007	^COMPLEX?	( → flag )
000007	^DEALMODE	Test complex mode. $(\rightarrow)$
09C007	^REALMODE	( → ) Set real mode, refresh configuration display.
09D007	^CLRCOMPLEX	Set real mode, refresh configuration display. ( $\rightarrow$ )
090007	CLRCOMPLEX	Set real mode.
09E007	^EXACTMODE	( $\rightarrow$ )
00007	пинстновы	Set exact mode, refresh configuration display.
09F007	^SETEXACT	$(\rightarrow)$
001007	SETEMICT.	Set exact mode and gcd mode.
0A0007	^NUMMODE	( → )
		Set numeric mode, refresh configuration dis-
		play.
0A1007	^CLREXACT	$(\rightarrow)$
		Clear exact mode.
0A2007	^EXACT?	( $ ightarrow$ flag )
		Test exact mode.

Addr.	Name	Description
0A3007	^STEPBYSTEP	$(\  ightarrow\ )$
		Set step by step flag, refresh display.
0A4007	^NOSTEPBYSTEP	$( \rightarrow )$
		Clear step by step flag, refresh display.
0A5007	^VERBOSEMODE	$( \rightarrow )$
		Set verbose mode, refresh configuration dis-
		play.
0A6007	^SILENTMODE	$( \rightarrow )$
		Set silent mode, refresh configuration dis-
		play.
0A7007	^RECURMODE	$( \rightarrow )$
		Set recursive mode, refresh configuration dis-
		play.
0A8007	^NONRECMODE	$( \rightarrow )$
		Set nonrecursive mode, refresh configuration
		display.
0A9007	^PLUSAT0	$( \rightarrow )$
		Set positive mode, refresh configuration dis-
		play.
0AA007	^SETPLUSAT0	$( \rightarrow )$
		Set positive mode.
0AB007	^PLUSATINFTY	$( \rightarrow )$
		Set positive infinity mode, refresh configura-
		tion display.
0AC007	^CLRPLUSAT0	$( \rightarrow )$
		Set positive infinity mode.
0AD007	^SPARSEDATA	$( \rightarrow )$
		Set full data mode, refresh configuration dis-
		play.
0AE007	^FULLDATA	$( \rightarrow )$
		Set sparse mode, refresh configuration dis-
		play.
0AF007	^RIGORMODE	$( \rightarrow )$
		Set rigorous mode, refresh configuration dis-
		play.
0B0007	^SLOPPYMODE	$( \rightarrow )$
		Set sloppy mode, refresh configuration dis-
		play.

Addr.	Name	Description
0B1007	^SLOPPY?	( $ ightarrow$ flag )
		Test sloppy mode.
1D2006	^SAVECASFLAGS	$( \rightarrow )$
		Saves CAS flags and current var.
1D4006	^RESTORECASFLAGS	$( \rightarrow )$
		Restore CAS flags and current var.
1D5006	^CASFLAGEVAL	$( \rightarrow )$
		Execute next runstream object with flag pro-
		tection.
0C2007	^RCLMODULO	$( \rightarrow Z )$
		Fetch MODULO from the home directory.
0C3007	^RCLPERIOD	( $ ightarrow$ sym )
		Fetch PERIOD from the home directory.
0C4007	^RCLVX	$( \rightarrow \text{id} )$
		Fetch VX from home directory.
0C5007	^STOVX	$( ob \rightarrow )$
		Store object in VX.
0C6007	^STOMODULO	$( ob \rightarrow )$
		Store object in MODULO.
0C7007	^RCLEPS	$( \rightarrow \% )$
		Fetch EPS from home directory.
0C8007	^ISIDREAL?	$(id \rightarrow id id T)$
		$(id \rightarrow id F)$
		Test if id is in the REALASSUME list.
0C9007	^ADDTOREAL	$(id \rightarrow )$
		Add idnt to the list of real var.
0CA007	^RESETCASCFG	$(\rightarrow)$
	_	Reset CAS config.
1D0006	^VERNUMext	( → %version )
		CAS version number.

# **Chapter 54 CAS Menus**

The entries in this chapter return the built-in menus of CAS commands, or do some other actions related to menus. For general information on menus, turn to Chapter 37.

### 54.1 Reference

Addr.	Name	Description
1D1006	^MENUXYext	( #2 #1 → {} )
		Make list of Erable commands between the
		given numbers.
08D007	^MENUext	( $\$6\$1 \rightarrow$ )
		If the CAS quiet flag is not set, displays the six
		strings as menu keys. Otherwise does nothing.
0B2007	^MENUCHOOSE?	( $ ightarrow$ prg flag )
		Return best CHOOSE command.
0B3007	^MENUCHOOSE	$(\ \{\}\  o\ )$
		Offers a selection to the user. If Flag -117
		is set, only installs a menu. If not, offer a
		CHOOSE box.
0B4007	^MENUGENE1	$( \rightarrow \{\} )$
		Menu for CAS.
0B5007	^MENUBASE1	$( \rightarrow \{\} )$
		Base algebra menu.
0B6007	^MENUCMPLX1	$( \rightarrow \{\} )$
		Complex operations menu.
0B7007	^MENUTRIG1	$( \rightarrow \{\} )$
		Trigonometric operations menu.
0B8007	^MENUMAT1	$( \rightarrow \{\} )$
		Matrix operations menu.

410 54. CAS Menus

Addr.	Name	Description
0B9007	^MENUARIT1	$( \ \rightarrow \ \{\} \ )$
		Arithmetic operations menu.
0BA007	^MENUSOLVE1	$( \rightarrow \{\} )$
		Solver menu.
0BB007	^MENUEXPLN1	$( \rightarrow \{\} )$
		Exponential and logarithmic operations menu.
0BC007	^MENUDIFF1	$( \rightarrow )$
		Differential calculus menu.

# **Chapter 55 Internal Versions of User RPL Commands**

The entries in this chapter are the closest correspondents to User  $\ensuremath{\mathsf{RPL}}$  commands.

### 55.1 Reference

Addr.	Name	Description
218006	^ISPRIME	$(z/\% \rightarrow \%0/\%1)$
		Internal ISPRIME.
1D6006	^FLAGEXPAND	$(symb \rightarrow symb')$
		Internal xexpand. Expands symbolic expres-
		sion.
1D8006	^FLAGFACTOR	$(symb \rightarrow symb')$
		( $z \rightarrow \text{symb}$ )
		Internal xFACTOR. Factors symbolic or num-
		ber.
1D9006	^FLAGLISTEXEC	( symb $\{\}  o $ symb' )
		Internal xSUBST for the case that level 1 is
		an array or a matrix.
1DA006	^FLAGSYMBEXEC	( symb symb' $ ightarrow$ symb'' )
		Internal xSUBST for the case that level 1 is a
		symbolic.
1DB006	^FLAGIDNTEXEC	( symb id $ o$ symb' )
		Internal xSUBST for the case that level 1 is
		an id or a lam.
1DC006	^FLAGINTVX	( $symb \rightarrow symb'$ )
		Internal xINTVX.

Addr.	Name	Description
1DD006	^DERVX	( $symb \rightarrow symb'$ )
		Internal xDERVX.
1DE006	^SOLVEXFLOAT	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
		Internal xSOLVEVX for a float.
1DF006	^SYMLIMIT	( symb symb' $\rightarrow$ symb'')
		Internal XLIMIT for scalars.
1E0006	^FLAGMATRIXLIMIT	$( [] symb \rightarrow []' )$
		Internal XLIMIT for matrices.
1E1006	^TAYLOR0	$(symb \rightarrow symb')$
		Internal xTAYLOR0.
1E2006	^FLAGSERIES	( symb id z $ ightarrow$ {} symb' )
		Internal xSERIES.
1E4006	^PLOTADD	( symb $ ightarrow$ )
		Internal xPLOTADD.
1E5006	^FLAGIBP	( $symb1 symb2 \rightarrow symb3 symb4$ )
		Internal xIBP.
1E6006	^FLAGPREVAL	( symb1 symb2 symb3 $ ightarrow$ symb4 )
		Internal xPREVAL. Evaluates symb1 at the
		points symb2 and symb3 and takes the dif-
		ference.
1E7006	^MATRIXRISCH	( [] id $ o$ symb')
		Internal xRISCH for matrix arguments.
1E8006	^FLAGRISCH	( symb id $ ightarrow$ symb' )
		Internal xRISCH for non-matrix argumetns.
1E9006	^FLAGDERIV	( symb id $ ightarrow$ symb' )
		Internal xDERIV.
1EA006	^FLAGLAP	( $symb \rightarrow symb'$ )
		Internal xLAP.
1EB006	^FLAGILAP	( $symb \rightarrow symb'$ )
		Internal XILAP.
1EC006	^FLAGDESOLVE	( symb symb' $ ightarrow$ symb'' )
		Internal xDESOLVE.
1ED006	^FLAGLDSSOLV	( $symb1 symb2 \rightarrow symb3$ )
		Internal xLDEC.
1EF006	^FLAGTEXPAND	$(symb \rightarrow symb')$
		Internal xTEXPAND.
1F0006	^FLAGLIN	(symb → symb')
		Internal xLIN.

Addr.	Name	Description
1F1006	^FLAGTSIMP	( symb $ o$ symb' )
		Internal xTSIMP.
1F2006	^FLAGLNCOLLECT	$(symb \rightarrow symb')$
		Internal xLNCOLLECT.
1F3006	^FLAGEXPLN	( $symb \rightarrow symb'$ )
		Internal xEXPLN.
1F4006	^FLAGSINCOS	( $symb \rightarrow symb'$ )
		Internal xSINCOS.
1F5006	^FLAGTLIN	( $symb \rightarrow symb'$ )
		Internal xTLIN.
1F6006	^FLAGTCOLLECT	$(symb \rightarrow symb')$
		Internal TCOLLECT.
1F7006	^FLAGTRIG	( $symb \rightarrow symb'$ )
		Internal xTRIG.
1F8006	^FLAGTRIGCOS	$(symb \rightarrow symb')$
		Internal xTRIGCOS.
1F9006	^FLAGTRIGSIN	$(symb \rightarrow symb')$
		Internal xTRIGSIN.
1FA006	^FLAGTRIGTAN	$( \text{symb} \rightarrow \text{symb'} )$
		Internal xTRIGTAN.
1FB006	^FLAGTAN2SC	$(symb \rightarrow symb')$
		Internal xTAN2SC.
1FC006	^FLAGHALFTAN	$( symb \rightarrow symb' )$
		Internal xHALFTAN.
1FD006	^FLAGTAN2SC2	$( \text{symb} \rightarrow \text{symb'})$
		Internal xTAN2SC2.
1FE006	^FLAGATAN2S	$( \text{symb} \rightarrow \text{symb'})$
		Internal xATAN2S.
1FF006	^FLAGASIN2T	$(symb \rightarrow symb')$
		Internal xASIN2T.
200006	^FLAGASIN2C	$(symb \rightarrow symb')$
		Internal xASIN2C.
201006	^FLAGACOS2S	$(symb \rightarrow symb')$
		Internal xACOS2S.
206006	^STEPIDIV2	$(z1 z2 \rightarrow z3 z4)$
		Internal xIDIV2.
207006	^FLAGDIV2	( $symb1 symb2 \rightarrow symb3 symb4$ )
		Internal xDIV2.

Addr.	Name	Description
208006	^FLAGGCD	( symb1 symb2 $\rightarrow$ symb3 )
		Internal xGCD for the case with two symbol-
		ica arguments.
209006	^PEGCD	( symb1 symb2 $\rightarrow$ symb3 symb4 symb5
		)
		Internal xEGCD for polynomials.
20B006	^ABCUV	( symb1 symb2 symb3 $ ightarrow$ symb4 symb5
		)
		Internal polynomial xABCUV.
20C006	^IABCUV	( z1 z2 z3 $\rightarrow$ z4 z5 )
		Internal integer xIABCUV.
20D006	^FLAGLGCD	( $\{\}  ightarrow \{\}$ symb )
		Internal xLGCD.
20E006	^FLAGLCM	( $symb1 symb2 \rightarrow symb3$ )
		Internal xLCM.
20F006	^FLAGSIMP2	( $symb1 symb2 \rightarrow symb3 symb4$ )
		Internal xSIMP2.
210006	^FLAGPARTFRAC	$(symb \rightarrow symb')$
		Internal xPARTFRAC.
211006	^FLAGPROPFRAC	$(symb \rightarrow symb')$
		Internal xPROPFRAC.
212006	^FLAGPTAYL	$(P(X) r \rightarrow P(X+r))$
		Internal xPTAYL.
213006	^FLAGHORNER	( symb1 symb2 $\rightarrow$ symb3 symb4 symb5
		)
		Internal xHORNER.
214006	^EULER	$(z \rightarrow z')$
		Internal xEULER.
216006	^FLAGCHINREM	( A1 A2 $\rightarrow$ A3 )
		Internal xCHINREM.
217006	^ICHINREM	( A1 A2 $\rightarrow$ A3 )
		Internal xICHINREM.
219006	^SOLVE1EQ	( symb id $\rightarrow$ $\{\}$ )
		Internal xSOLVE for single equations.
21A006	^SOLVEMANYEQ	$( [] []' \rightarrow \{\}'')$
		Internal xSOLVE for arrays of equations.
21B006	^ZEROS1EQ	( symb id $\rightarrow$ {} )
		Internal xZEROS for single equations.

Addr.	Name	Description
21C006	^ZEROSMANYEQ	$( \hspace{.1cm} [\hspace{.1cm}] \hspace{.1cm} [\hspace{.1cm}] \hspace{.1cm}   \hspace{.1cm} \rightarrow \hspace{.1cm} \{\hspace{.1cm}\} \hspace{.1cm})$
		Internal xZEROS for arrays of equations.
21D006	^FCOEF	( [] $ ightarrow$ symb )
		Internal xFCOEF.
21E006	^FROOTS	( symb $ ightarrow$ [])
		Internal xFROOTS.
21F006	^FACTORS	( symb $ ightarrow$ {} )
		Internal xFACTORS.
220006	^DIVIS	( $symb \  o \ \{\}$ )
		Internal xDIVIS.
223006	^rref	( M $\rightarrow$ A M' )
		Internal xrref.
229006	^MADNOCK	( M $ ightarrow$ symb1 []' []'' symb3 )
		Internal xMAD.
22A006	^SYSTEM	( [] []' $ ightarrow$ []'' $\{\}$ []''' )
		Internal xLINSOLVE.
22B006	^VANDERMONDE	( $\{\} \rightarrow M$ )
		Internal xVANDERMONDE.
22C006	^HILBERTNOCK	$(z \rightarrow M)$
		Internal xHILBERT.
22E006	^CURL	( [exprs] [vars] $ ightarrow$ [] )
		Internal xCURL.
22F006	^DIVERGENCE	( [exprs] [vars] $ ightarrow$ symb )
		Internal xDIV.
230006	^LAPLACIAN	( [expr] [vars] $ ightarrow$ symb )
		Internal xLAPL.
231006	^HESSIAN	( symb A $ ightarrow$ M A' A'' )
		Internal xHESS.
232006	^HERMITE	( $z \rightarrow symb$ )
		Internal xHERMITE.
233006	^TCHEBNOCK	( $ ext{ } ext{degree}  ightarrow  ext{symb}$ )
		Internal xTCHEBYCHEFF.
234006	^LEGENDRE	( $z \rightarrow symb$ )
		Internal xLEGENDRE.
235006	^LAGRANGE	( A $ ightarrow$ symb )
		Internal xLAGRANGE.
236006	^FOURIER	$(symb_z \rightarrow C%)$
		Internal xFOURIER.

Addr.	Name	Description
238006	^TABVAR	( $symb \to symb \ \{\{\}\} \ grob$ )
		Internal xTABVAR.
239006	^FLAGDIVPC	( symb1 symb2 z $ ightarrow$ symb3 )
		Internal xDIVPC.
23A006	^FLAGTRUNC	( $symb1 symb2 \rightarrow symb3$ )
		Internal xTRUNC.
23B006	^FLAGSEVAL	$(symb \rightarrow symb')$
		Internal xSEVAL.
23C006	^XNUM	( $symb \rightarrow symb'$ )
		Internal xXNUM.
23D006	^REORDER	( symb id $ ightarrow$ symb' )
		Internal xREORDER.
23E006	^USERLVAR	$( symb \rightarrow symb [] )$
		Internal xLVAR.
23F006	^USERLIDNT	$( symb \rightarrow [] )$
		Internal xLNAME.
241006	^ADDTMOD	( $symb1 symb2 \rightarrow symb3$ )
		Internal xADDTMOD for scalars.
242006	^MADDTMOD	( M M' $\rightarrow$ M'' )
		Internal xADDTMOD for matrices.
243006	^SUBTMOD	( $symb1 symb2 \rightarrow symb3$ )
		Internal xSUBTMOD for scalars.
244006	^MSUBTMOD	$(MM' \rightarrow M'')$
		Internal xSUBTMOD for matrices.
245006	^MULTMOD	$(symb1 symb2 \rightarrow symb3)$
		Internal xMULTMOD.

# **Chapter 56 Miscellaneous**

In this chapter are listed the entries that did not fit in any of the previous chapters.

### 56.1 Reference

### 56.1.1 Verbose Mode Display Routines

Addr.	Name	Description
579006	^Verbose1	<b>(</b> \$ → )
		Display message on line 1 if verbose mode on.
57A006	^Verbose2	$(\ \ \ \ \ \ \ \ \ \ \ )$
		Display message on line 2 if verbose mode on.
57B006	^Verbose3	$( \ \ \ \ \ \ \ \ \ \ \ )$
		Display message on line 3 if verbose mode on.
57C006	^VerboseN	$( \ \$ \ \# \rightarrow \ )$
		Display message on given line if verbose mode on.

#### 56.1.2 Evaluation

Addr.	Name	Description
257006	^EvalNoCKx*	( ob ob' $ ightarrow$ ob'' )
258006	^EvalNoCKx+	( ob ob' $ ightarrow$ ob'' )
259006	^EvalNoCKx-	( ob ob' $ ightarrow$ ob'' )
25A006	^EvalNoCKx/	( ob ob' $ ightarrow$ ob'' )
25B006	^EvalNoCKx^	( ob ob' $ ightarrow$ ob'' )
25C006	^EvalNoCKxCHS	( ob $ ightarrow$ ob' )
25D006	^EvalNoCKxINV	( ob $ ightarrow$ ob' )
25E006	^EvalNoCKxMOD	( ob ob' $\rightarrow$ ob'' )

418 56. Miscellaneous

Addr.	Name	Description
25F006	^EvalNoCKxPERM	( ob ob' → ob'' )
260006	^EvalNoCKxCOMB	( ob ob' $\rightarrow$ ob'' )
261006	^EvalNoCKxOR	( ob ob' $\rightarrow$ ob'' )
262006	^EvalNoCKxAND	( ob ob' $ ightarrow$ ob'' )
263006	^EvalNoCKxXOR	( ob ob' $\rightarrow$ ob'' )
264006	^EvalNoCKxXROOT	( ob ob' $\rightarrow$ ob'' )
265006	^TABVALext	( fnct x $\{\} \rightarrow \{\}'$ )
		Table of values.

### 56.1.3 Conversion

Addr.	Name	Description
266006	^TOLISText	( o1on $\#n \rightarrow Lvar Q1Qn$ )
		Convert meta of symbolic objects to internal
		form.
267006	^FROMLISText	( Lvar Meta L $ ightarrow$ L' )
		Conversion of elements of Meta objec to user
		format. Meta does not contain the #n number
		of element. L is the list of depth of the elements
		of Meta. For example to convert a polynomial,
		a vector and a matrix:
		Lvar = { X }
		$Meta = \{ Z1 Z3 \}$
		{ Z0 Z1 }
		{ { Z1 { Z1 Z0 } } }
		L = { #0 #1 #2 }
		$L' = \{ 'X+2' \{ 0 1 \} \{ \{ 1 X \} \} \}.$

# 56.1.4 Qpi

Addr.	Name	Description
074007	^QPI	( $ob \rightarrow ob'$ )
		Internal xXQ.
073007	^QpiZ	( $ob \rightarrow symb$ )
		Calls ^Qpi% and converts the resulting (real) inte-
		gers into zints.

Addr.	Name	Description
075007	^QpiSym	( $symb \rightarrow symb'$ )
		Internal xXQ for symbolics.
076007	^QpiArry	$( [] \rightarrow []' )$
		Internal xXQ for arrays. Converts each element of
		the array.
077007	^QpiList	$(\ \{\}\  o\ \{\}'\ )$
		Internal xXQ for lists. Converts each element of the
		list.
078007	^Qpi	( $%/C% \rightarrow \text{symb}$ )
		Internal $xxQ$ for real and complex numbers.
079007	^Qpi%	( % $ ightarrow$ symb )
		xXQ for reals, but does not convert numbers to zints.
07A007	^GetRoot	( %' → %' %'')
		Tries to find a square number which is a factor
		of the argument. The algorithm only tries num-
		bers smaller than 1024^2-1 and assumes that %
		is an integer. The returned results are such that
		$=(\%')^2*\%''$ . For numbers which do not contain a
		square factor, $\%'=1$ and $\%''=\%$ .
07B007	^Approx	( % -> %' %'')
		Approximates a real number with a fraction. Re-
		turns numerator %' and denominator %". The ac-
		curacy of the approximation is determinated by the
		current display format.

# **56.1.5** Infinity

_

420 56. Miscellaneous

Addr.	Name	Description
2E7006	^POSUNDEFext	( symb $ o$ symb $\#$ )
		Returns #1 if the symbolic contains the unde-
		fined symbolic '?'.

### 56.1.6 Built-In Constants

Addr.	Name	Description
2EA006	^pi	$( \rightarrow \ '\pi' \ )$
2EB006	^metapi	( $\rightarrow$ $\pi$ #1 )
2F1006	^meta-pi	( $\rightarrow$ $\pi$ xNEG #2 )
2E8006	^pisur2	$( \rightarrow '\pi/2' )$
2F2006	^metapi/2	( $\rightarrow$ $\pi$ 2 x/ #3 )
2E9006	^pisur-2	$( \rightarrow '-\pi/2')$
2F4006	^meta-pi/2	( $\rightarrow$ $\pi$ 2 x/ xNEG #4 )
2F3006	^metapi/4	( $\rightarrow$ $\pi$ 4 x/ #3 )
2F5006	^meta-pi/4	( $ ightarrow$ $\pi$ 4 x/ xNEG #4 )
2F6006	^pifois2	$( \rightarrow '2*\pi')$
2EC006	^'xPI	$( \rightarrow xPI )$
2F9006	^base_ln	$( \rightarrow \ \ \ \ )$
2FA006	^meta_e	( $\rightarrow$ $e$ #1 )
2EE006	^'xi	$( \rightarrow xi )$
2ED006	^metai	$(\rightarrow i \# 1)$
2EF006	^ipi	$( \rightarrow ' i * \pi' )$
2F0006	^metaipi	( $\rightarrow$ i $\pi$ x* #3 )
2F8006	^metapi*2	( $\rightarrow \pi$ 2 x* #3 )
2F7006	^deuxipi	$( \rightarrow '2*i*\pi')$

### 56.1.7 List Application

Addr.	Name	Description
3F0006	^DIVOBJext	( $\{o1on\}$ ob $\rightarrow$ $\{o1/obon/ob\}$ )
		Division of all elements of a list by ob. Tests if
		ob=1.
3F2006	^LOPDext	$( \{o1on\} ob \rightarrow \{o1/obon/ob\} )$
		LOPDext calls QUOText for the division, unlike
		DIVOBJ which calls RDIVext.

Addr.	Name	Description
269006	^LOP1ext	<pre>( { } ob binop → { } ' ) Applies non-recursively &lt;&lt; ob binop &gt;&gt; to the elements of the list.</pre>
26A006	^LOPAext	( $\{\}$ ob binop $\to$ $\{\}$ ') Applies recursively $<<$ op binop $>>$ to the elements of the list (not the list elements them-
10F006	^LOPMext	selves). ( ob {} → {}' ) Multiplies each element of the list by the given object.
45F006	^LISTEXEC	( ob $\{\} \rightarrow$ ob' ) ( ob $\{\} \rightarrow$ $\{\}$ ' ) The list should be of the form $\{$ 'X=1' 'Y=2' $\}$ in the first case or $\{$ 'X=1' 'X=2' $\}$ in the second case. In the first case, all occurences of X in ob are replace by 1, or Y by 2, etc. In the second case ob is evaluated with X=1, X=2 successively.
460006	^LISTEXEC1	( {} objet → {}' )
461006	^SECOEXEC	( $\{\} \text{ prog } \rightarrow \{\} \}$ ) Executes prog on each element of ob.
268006	^PFEXECext	$($ symb prg $\rightarrow$ symb $)$
26B006	^LISTSECOext	( composite $\rightarrow$ composite ) Applies 1LAM non-recursively to all elements of the list.
26D006	^CK1TONOext	( ob $\rightarrow$ ob' ) Applies prg to ob, recursively for lists. prg is fetched from runstream.

# 56.1.8 Irrquads

Addr.	Name	Description
167006	^TYPEIRRQ?	( ob $ ightarrow$ flag )
		Is ob an irrquad?
168006	^DTYPEIRRQ?	( ob $ ightarrow$ ob flag )
		DUP, then ^TYPEIRRQ?.
165006	^QXNDext	( $irrq \rightarrow a b c$ )
		b=0 and c=1 if stack level 1 is not an irrg.

422 56. Miscellaneous

Addr.	Name	Description
166006	^NDXQext	( a b c $\rightarrow$ irrq )
2D8006	^IRRQ#ULTIMATE	$( ob \rightarrow \# c )$
		Finds « depth and returns ultimate c of an irrq.
508006	^QCONJext	( $irrq \rightarrow irrq'$ )
		irrq-conjugate of an irrq. This is <i>not</i> the com-
		plex conjugate.
509006	^QABSext	( $irrq \rightarrow irrq sign$ )
		Finds the sign of an irrq. Work always if irrq is
		made of Z.
51A006	^QNORMext	( Zirr $\rightarrow$ a^2-b*c^2 )
		Irrq-norm of an irrquad. This is <i>not</i> the com-
		plex modulus.
4D4006	^SECOSQFFext	( :: x<< a b c x>> $\rightarrow$ { fact1 mult1
		<pre> factn multn } )</pre>
		Factorization of irrquads and Gauss integers.
124006	^PREPARext	( o1 o2 $\rightarrow$ a1 b1 c1 a2 b2 c2 )
		Returns irrquad decomposition of o1 and o2.
		with either c1=c2 or c1 and c2 have no fac-
		tors in comon. c1 <c2, by<="" handled="" ordering="" td=""></c2,>
		LESSCOMPLEX? is made by type, then by CRC.
2DA006	^LISTIRRQ	( ob $\{\} \rightarrow \{\}'$ )
		Add the C-part of all irrquads of object to the
		list.

### 56.1.9 Miscellaneous

Addr.	Name	Description
3E7006	^PSEUDOPREP	( o2 o1 $\rightarrow$ o2*a1.n^ o1 a1.n^ )
3FB006	^HSECO2RCext	( ob $ ightarrow$ ob' )
		Conversion of constants from internal to user
		form.
3FC006	^SECO2CMPext	( seco $ ightarrow$ symb )
		Back conversion of complex. polarflag should
		be disabled if not at the top level of rational
		expressions.

Addr.	Name	Description
3FF006	^VALOBJext	( # $\{\{Q\}\}$ {var1varn} $\rightarrow$ $\{\{ob\}\}$ ) Back conversion of objects embedded at depth # in lists. Simplifies var1varn.
401006	^VAL2ext	( # {{Q}} {var1varn} → {{ob}} )  Back conversion of objects embedded at depth # in lists. Does not simplify var1varn. Conversion is done in asc. power if positivfflag is set, which is useful for SERIES and LIMIT commands.
402006	^INVAL2	( P # $ ightarrow$ symbpoly ) LAM2 must contain Lvar, # is the depth.
403006	^METAVAL2	( $\#$ Meta_list $ o$ Meta_symb ) LMA2 must contain Lvar, LAM1 is modified.
404006	^VAL1	( ob $ ightarrow$ ob ) LAM2 must contain Lvar, LAM1 is modified.
405006	^VAL1M	<ul> <li>( ob → Meta_symb )</li> <li>LAM2 must contain Lvar, LAM1 is modified.</li> </ul>
45C006	^IDNTEXEC	( symb idnt → symb') Tries to find idnt such that symb=0. Return a solution as an equality 'idnt=' in symb'.
121006	^MP0	( $ob \rightarrow ob 1$ ) Returns number 1 of the selected type. The symbolic/ROMPTR one looks very strange it is used to avoid infinity $outletoonup ode over the symbol of outletoout of outlet$
26C006	^rpnQOBJext	( ob → ob' ) prg is fetched from the stack. Looks for all d1, d2, at the beginning of the name of idnt to determine if idnt represents a derivative of a user function. Stops if at a time the stripped idnt is in the current directory. Example   'd2d1Y' returns { #2 } << >> if 'd2d1Y' is not defined and 'd1Y' is defined as << >> or   { #2 #1 } 'Y' if d2d1Y d1Y and Y are not defined.

424 56. Miscellaneous

Addr.	Name	Description
29D006	^SIMPIDNT	( idnt $\rightarrow$ ob )
		Evaluates idnt (looks recursively for its con-
		tent if defined). Does not error for circular
		definition, but displays a warning.
29F006	^RCL1IDNT	( $idnt/lam \rightarrow ob$ )
		Recursive content of an idnt. LAM1 to
		LAM3 must be bound.
2A7006	^SWPSIMPNDXF	( ob2 ob1 $\rightarrow$ ob1/ob2 )
		Simplified fraction (internal).
2A8006	^SIMPNDXFext	( ob2 ob1 $\rightarrow$ ob2/ob1 )
		Simplified fraction (internal).
2B6006	^CMODext	( C2 C1 $\rightarrow$ C1 C2_mod_C1 )
2BD006	^SQFF2ext	( l1ln $\#$ n-1 $\rightarrow$ l1'ln' $\#$ n-1 )
2BE006	^PPZ	( p $\rightarrow$ p/pgcd pgcd )
		ob is the gcd of all constant coefficients of P
		(integer, Gauss integers, irrquads with the
		implementation of the "gcd" for irrquads).
117007	^PPZZ	( ob $ ightarrow$ ob zint )
		PPZ with further check to ensure returning a
		zint.
2BF006	^PZHSTR	( a z $\rightarrow$ a mod z )
2C0006	^HORNER1ext	$(Pr \rightarrow P[r])$
2C1006	^PEval	$(Pr \rightarrow P[r])$
		P must be a list polynomial.
2C6006	^SQRT_IN?	$(\ \{\}\  o\ \{\}\  ext{flag}\ )$
		Returns TRUE if one element of {} is a symb
		containing a sqrt.
2C7006	^IS_SQRT?	( $symb \rightarrow flag$ )
2C9006	^IS_XROOT?	( $symb \rightarrow flag$ )
2CA006	^STOPRIMIT	$(symb \rightarrow )$
		Stores antiderivative in PRIMIT variable.
2CB006	^CONTAINS_LN?	( $symb \rightarrow symb flag$ )
2D4006	^FOURIERext	( symb n $\rightarrow$ cn )
		Computes n-th Fourier coefficient of a 2 $\pi$ pe-
		riodic function.

Addr.	Name	Description
2D9006	^LESSCOMPLEX?	( ob1 ob2 → ob1 ob2 flag ) Compares objects by type and then by CRC. flag is true if ob1 is less complex than ob2 (ob1>ob2). If ob1 or ob2 is an irrq, find first ultimate type of ob1 and ob2. If these ultimate types are equal sort is done by comparing the << depth.
2DD006	^TABLECOSext	( $\rightarrow$ {} ) Table of special COS values (k*pi/12).
2DE006	^TABLETANext	$( \rightarrow \{ \} )$ Table of special TAN values (k*pi/12).
101007	^LINEARAPPLY	( symb nonrat_prg rat_prg $\rightarrow$ symb ) Applies linearity. nonrat_prg is applied for a non rational part symb $\rightarrow$ symb. rat_prg is applied for a rational part symb $\rightarrow$ symb. Linearity is applied on symb.
106007	^A/B2PQR	( A B $\rightarrow$ P Q R ) Writes a fraction A/B as E[P]/P*Q/E[R]. Q and positive shifts of R are prime together.
107007	^GOSPER?	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$
0CB007	^FRACPARITY	( fr $\rightarrow$ Z ) Tests if a fraction (internal rep) is even/odd/none. Z=1 if even, -1 if odd, 0 if neither even nor odd.
0D5007	^FR2ND%	( fraction-l $\rightarrow$ N D % ) Extract trivial power of fraction.
4D1006	^MSECOSQFF	( ob $ ightarrow$ Meta ) Factorization of an extension.

# Part V

# Appendices

# Appendix A Development Tools

You have basically two choices for developing software for the HP49G. The programs can either be written and tested on a PC, using special tools and an emulator, or you can write software directly on the HP49G.

This chapter will describe tools for the HP49G calculator that make it a suitable programming environment for System RPL development. The HP49G calculator includes a built-in compiler, disassembler and some sort of debugger (which, to say the truth, could be improved), plus some other little tools that can be of use to the System RPL programmer. However, for big programming tasks this is not enough: some other tools are necessary to make programming easier. Because of this, some third-party tools will also be described. With a good knowledge of the built-in and third-party tools, the HP49G can be used as a complete and compact programming environment. All the programs described here can be freely downloaded from The HP Software Archive, http://www.hpcalc.org.

The built-in programming tools you will need are, by default, not accesible to the user. They are in two libraries, which are not attached by default. Library 256 contains several useful commands for "hacking" with the calculator, and also the disassembler. Library 257 contains MASD, the compiler. You should have these libraries always attached. If you have extable installed (and you should — see section A.1), then library 256 will be automatically attached. Library 257 (MASD) does not really need to be attached, because it is possible to call MASD from library 256. Nevertheless, it is still good to have it attached.

The STARTUP variable is useful to configure the calculator. This variable (which must be in the HOME directory) contains an object to be executed after each warmstart. It can be used to set all parameters lost by a warmstart that you want to keep, or to do anything else you want. The following program will set user mode (which is lost in a warmstart); for efficient programming (and even for efficient use) it is essencial to make some key assignments. The program also attaches library 257.

<sup>« -62</sup> SF 257 ATTACH »

### A.1 The Entry Points Library

For System RPL development, the extable library is virtually indispensable. This library contains the tables of supported entry points and addresses. It is with the help of this library that you can write DUP and get the correct address for this command; without it, you would need to enter PTR 3188 every time or write an equate for this command manually. In disassembly (including the System RPL stack (see section A.3)), it allows you to get the name of the commands, instead of only their addresses. Basically, this library is pretty much essential.

Transfer extable to your calculator and install it as any other library. That is all you need to do to use command names instead of addresses. Extable appears in the library menu, and it contains five user-accessible commands.

The first command, nop, does nothing:-). Probably, there was a command in that position before, but it was removed, and another command that does nothing was put there not to change the other rompointers.

The other four commands, fortunately, are sometimes useful :-) (if not directly then through the Emacs library). The GETADR command returns the address of an entry. Just put the name of the entry (a string) in level one and run it. The inverse operation is done by GETNAME: give it an address, and it will return the name of the entry.

If you do not know the exact name of an entry, the last two commands will help you. Put a string with the first few letters of the command in level one, run GETNAMES and, voilà, a list with the names of all commands that *start* with those letters is returned. The last command, GETNEAR, is even more powerful: give it a string, and all commands whose names *contain* that string (even if in the middle of the command) will be returned.

# A.2 About Key Assignments

Even though assigning keys is not directly related to System RPL programming, we will describe here the KEYMAN library, written by Wolfgang Rautenberg (e-mail: raut@math.fu-berlin.de). The latest version is 9.2001. This library simplifies the assignment, deletion and recalling of keys, but, most importantly, allows a key to behave differently if it is pressed longer than usual or double pressed.

You will find several commands inside this library. The A?D command

is used to assign and delete keys. To assign something to a key, put the object in level one and press A?D shortly. Then, press the key you want to assign to (shifts and shift-holds work, of course). The key is assigned. To delete an assignment, press A?D for a slightly longer time, and then the key from which you want to remove the assignment. The command Rclk allows one to recall the assignment of any key. It works like the previous commands: press it (briefly) and then the key. A longer press will return a list of all the keys assigned.

The commands above are just other ways to do what was already possible with the built-in commands. But the real power is in the IfE?P, IfD and IfL commands. The first serves two functions: it allows a key to have different meanings when in edit mode and when not, or to have different meanings when in program mode and when not. To use it, put the object to be run in edit or program mode in level two, the object to be run in normal mode in level one, and press IfE?P. A short press will create a program that evaluates the object in level two if the calculator is in edit mode, or the object in level one if not. A longer press does the same, but the test is based on whether program entry mode is active or not.

The IfD and IfL commands are similar. To use IfD, put in level two the object to be run if the key is pressed twice (like with a computer mouse) — double pressed — and put in level one the object to be run if the key is pressed once. Run IfD, and you will have a single program that executes one of the objects according to how the key was pressed. Note that assignments produced with IfD will slightly delay execution on a single keypress, since the calculator must wait to see if the double press will happen or not. The command IfL is similar, but it allows different actions based on how long the key is pressed: you have seen this behaviour in the A?D command. The object to be run in a longer press is in level two.

All the If commands have an extra feature. Any of the two objects in the stack can be a real number in the form rc.p, where r is the row, c is the column and p is the plane (normal, left-shifted, right-shifted, left-shift-hold, etc.). In the program created with the If commands, these numbers will be replaced by the standard key assignment of the corresponding key. This is really useful for making assignments which do not disturb the normal function of a key but just add functionality in a special mode or keypress technique. If you give a real number that is not a valid keycode, it will be replaced by a command to make a beep.

Two other commands can sometimes be useful: →TO? inserts the System RPL command TakeOver in the beginning of the program when the key

is pressed shortly. This is necessary if you want the command to be executed while the command line is active. A longer press inserts UnlockAlpha in the beginning of the program, useful when it is assigned to an alpha-shifted key. Finally, K&SA recalls the keycode and standard assignment for any key. This is used when you want to add new functionality to a key. When this standard assignment is a command in a library (that is, a ROM Pointer, also called a XLIB name), the pointer is recalled to level two, and its contents is put in level one.

In the following sections, we will show some examples of key assignments built with KEYMAN commands.

### A.3 Hacking Tools

The tools described here make the life of the programmer easier. They give access to some functions which are normally not available for pure User RPL users of the calculator. First, the built-in tools in the HP49G will be described. Later, a third-party library will be described.

Before describing the built-in tools found in library 256, we will mention a flag that is very useful to System RPL programmers: flag –85. When this flag is set, the "System RPL Stack" is active: in the stack the objects are decompiled using the System RPL decompiler before being displayed. That means that, where one would see just External with the normal stack, the name for the entry (or PTR and the address, if no name is found) will be displayed, if you have the extable library (see section A.1) installed. Play with it a bit and you will see how useful it can be. Some objects (most notabily real numbers and integers) keep their usual notation, but in the interactive stack all objects are decompiled. This "System RPL Stack" is like the one produced by the command SSTK command of the JAZZ library for the HP48 calculators.

Probably you will be switching between the two kinds of stack display all the time. It is a good idea to assign a simple program to a key to toggle this display. A possibility is to assign it to Right-shift MODE. This normally is the key that marks the end of selection in edit mode. Since this key is unused when not in edit mode, it is a good example of the use of the KEYMAN library. To create this assignment, first put the program to be run when in edit mode in level two. This is easy: just use keycode 22.3. Then, write a simple User RPL (or System RPL, if you want) program to toggle flag −85 (this task if left to the reader — but read the description of OT49 in section A.3.1 first). Finally, press IfE?P briefly and use →TO? on the resulting program (because it must

be able to run while in edit mode), and assign it to the key, with A?D or the ASN command.

Library 256 contains some useful tools for the programmer. This library does not show up in the library menu (because it does not have a title), but you can get a menu with its commands by typing 256 MENU. If the library is attached (as it should be), you can type the commands, look up them in the catalog and an option will appear in the Apps menu, which says "Development lib", giving access to all the commands in the library.

Here is a description of the commands present in the library:

Command	Description
$\rightarrow$ H	"To hex": This converts an object into a string of hexadecimal
	characters. A common tool since the HP48 days to ease trans-
	fer of binary objects.
${\tt H}{\longrightarrow}$	"From hex": This is the opposite transformation: creates an
	object from a string of hexadecimal characters.
ightarrowA	"To address": Given an object, this command returns the ad-
	dress of the object, which is always a five-nibble hxs. Objects
	whose address is less than # 80000h are in ROM, and objects
	whose address is greater than that are in RAM.
$\mathbb{A}{\longrightarrow}$	"From address": This recalls the object at the specified ad-
	dress.
$S {\longrightarrow} H$	"String to hex": Converts a string into its characters' hexadec-
	imal representation. For example, since 5A, 59 and 58 are the
	hexadecimal codes for X, Y and Z respectively, "XYZ" becomes
	"8595A5".
${\tt H} {\rightarrow} {\tt S}$	"Hex to string": The opposite transformation.
$ ightarrow  exttt{LST}$	"Make list": Creates a list from a user meta object or another
	composite. A user meta object is any number of objects in the
	stack followed by a count represented as a real number. Be
	careful, because this command is not sufficiently argument-
	protected.
ightarrowALG	"Make algebraic": Creates an algebraic object from a user
	meta object or another composite. This may easily result in
	'Invalid Expression'.
$\rightarrow$ PRG	"Make program": Creates a program from a user meta object
	or another composite.
$COMP \rightarrow$	"From composite": Explodes any composite object into a user
	meta object.

Command	Description
$\rightarrow$ RAM	"To RAM": Dumps any ROM object into RAM. Can extract
	some commands for disassembly, but see section A.5 for more
	information.
SREV	"Reverse string": Reverses a string, very fast.
POKE	Writes data to any address in RAM. Put in level two a hxs
	with the address, and in level one a string of hex digits to be
	written at that address. This is a very easy way of destroying
	any "masterpiece" you have created on the calculator :-).
PEEK	Extracts raw hex digits from any address. Put the address in
	level two (an hxs) and the number of nibbles to get (another
	hxs) in level one.
APEEK	"Address peek": Like PEEK, but always gets five nibbles, re-
_ ~	turning them as a hxs.
R~SB	"Real ↔ system binary": Converts reals to bints and vice-
ap~p	"System himowy himowy". Convents hints to bye's and vise
SB~B	"System binary ↔ binary": Converts bints to hxs's, and vice-
LR~R	versa. "Long real ↔ real": Converts long reals to reals and vice-versa.
S~N	"String ↔ name": Converts strings to identifiers (global
S N	names) and vice versa.
LC~C	"Long complex   complex": Converts long complexes to com-
	plexes and vice-versa.
${\tt ASM}{\to}$	"From ASM": Disassembles Code objects (machine-language)
	into source code.
CRLIB	"Create library": A library creator. This is described in Ap-
	pendix B.
CRC	Calculates the CRC. The argument is a string of hex digits.
MAKESTR	"Make string": Creates a string with the number of characters
	given in level one (a real number).
SERIAL	Returns a string with the internal Serial Number of the HP49.
ASM	Provides access to the MASD compiler. See section A.4 for
	more information.
ER	Used in conjunction with ASM. See section A.4 for more infor-
	mation.
→S2	Disassembles an object. See section A.5 for more information.
XLIB~	Creates a rompointer (XLIB name) from the library number
	(level two) and command number (level one). It also explodes
	rompointers into its two components.

#### A.3.1 Operating Tools for the HP49

Wolfgang Rautenberg is the author of a library called Operating Tools (or OT49 for short) with several commands, some of which are useful to the System RPL programmer. The latest version of this library is 3.2002.

OT49 contains a library creator and a library splitter (written by Peter Geelhoed). To split any library, just put its number in the stack and run  $D \leftrightarrow L$ .

The DType command displays the type of the object in level one. If that object is a rompointer (XLIB) or flashpointer, its contents is recalled (unless it is pure machine-language code) and the contents' type is displayed, with an asterisk appended.

One of the most useful commands is <sup>3</sup>tog. It toggles between three representations of composite objects: as a list, as a program and as a user-metaobject. This can be used to manipulate System RPL programs without actually decompiling them. <sup>3</sup>tog explodes a program onto the stack, you can use stack commands to rearrange things and then <sup>3</sup>tog again to rebuild the program.

Another very useful command is F1~. It is a flag toggler. Just give the number of the system or user flag, run it, and the flag is toggled. It will also display in the header what has just been done.

The MDA~ command compiles or decompiles an object (depending wheter the input is a string or another object).

### A.4 The Compiler

The compiler included in the HP49G calculator is MASD. It is a newer version of the compiler found in the MetaKernel program for HP48G calculators. If you have already used the MetaKernel, then you can probably skip most of this section. But, even if you have never used MASD, there should be no difficulties learning how to use it. There are no big differences between MASD syntax and that of other System RPL compilers such as JAZZ (for the HP48 calculators), the HP Tools or the GNU Tools.

MASD is called with the command ASM. It expects a string in level one, and returns the compiled object. If there are errors, the string and a list will be put in the stack. This list is used by the ER command, described shortly.

The first difference to be observed from those that are coming from JAZZ or one of the PC Tools is that MASD, for some unknown reason, needs the

source to end with a "@" character. All source code files must be identified with this token, or MASD will refuse to even look at them. The character must be on a line by itself, at the start of the line, and with no character after it (not even a newline). This way, it is pretty much just cumbersome. (To be useful, it would be the character marking the end of the source, but there should not be all those restrictions on its placement, and text after it should be allowed — and ignored.) However, for the Emacs RPLCPL (see section A.6), the @ acquires at least one purpose: it allows the calculator to automatically distinguish between a System RPL source file and a Uer RPL program or command line.

The other thing to note concerns the current MASD *mode*. There are two modes, selected by flag –92: Assembly Language mode (flag –92 cleared) and System RPL mode (flag –92 set). Probably, you will set flag –92 and thus MASD will be by default in System RPL mode. Then, nothing else needs to be changed to compile System RPL programs (just add the @ in the end). It is still possible to compile Assembly Language code in System RPL mode: just surround the code between CODE and ENDCODE.

If you are in Assembly Language mode, it is possible to compile System RPL code by inserting these two lines before the source:

```
1 !NO CODE
!RPL
```

Both are called directives. The !NO CODE directive tells MASD to compile our source as System RPL code, and not as Machine Language code. (Once more, you can insert assembly language code between CODE and ENDCODE.) It is a good idea to always put these two lines at the start of all programs even if you use System RPL mode: this way, the source can be compiled regardless of the flag settings.

Here is a simple program source ready for MASD:

```
1 !NO CODE
    !RPL
    ::
        DUPTYPEZINT?
5     case
        FPTR2 ^Z>R
        DUPTYPEREAL? ?SEMI
        SETTYPEERR
    ;
10 @
```

The above is the disassembly of the CKREAL entry. As you can see, it automatically converts integers to real numbers.

It is a nice idea to assign the ASM command to a key: you will need it many times.

If there was an error during compilation, the original string is put in level two, and a list is put in level one. In this case, run the ER command. It will display a list of errors for you to choose, and will jump directly to that error in the source. Correct the error, press ENTER and the choose another error, until all errors have been corrected. Then, run ASM (and ER, if necessary) again. Better yet, use the ASM2 command from library 257, which calls ASM and then, if there was any error, ER.

#### A.4.1 MASD and the Different Kinds of Entries

A System RPL program can call three different kinds of entries: normal entries, which point to some address in ROM, flashpointer entries, which point to a command in one of the HP49's flash banks, and rompointer entries, which point to a command in a library (built-in or not).

For supported "normal" entries, no special precautions need to be taken. You can just include the name of the command. To call an unsupported entry, you will have to use PTR <address>, where <address> is the address as listed in the tables.

For supported flashpointer entries (whose names always start with ^), you have to prefix the entry's name with FPTR2. So, to call the flashpointer command ^Z>R, you will have to include this in your program:

FPTR2 ^Z>R

An unsupported flashpointer entry is called with FPTR <bank> <cmd>.<bank> are the last three digits of the address as listed in the table (but in practice it can be no bigger than Fh), and <cmd> are the first three digits.

Calling rompointer entries (which have names starting with ~) is very similar to calling flashpointers. If it is supported, just prefix it with ROMPTR2. For unsupported entries, you have to use this syntax: ROMPTR <lib> <cmd> cmd>.are the last three digits of the address, and <cmd> the first three.

### A.4.2 MASD's Special Features

The MASD compiler supports some special features that are not a part of the System RPL programming language, but that can be useful to the programmer.

The first feature eases the use of unsupported entries. You can define a name for a unsupported entry, making it behave as if an entry in extable. (This only works for normal entries, not flashpointer or rompointers.) To do that, use the following structure:

```
EQU name address
```

where name is the name of the entry, and address is its address. For example, the line below defines the entry <code>2NELCOMPDROP</code>, which returns the second element of a composite:

```
EOU 2NELCOMPDROP 2825E
```

With that definition, you can use 2NELCOMPDROP instead of PTR 2825E to access that command. Note that this only works for normal entries.

Another way to ease the inclusion of unsupported entries (especially rompointers and flashpointers), but that is useful not only for that, are the DEFINES. The structure is like this:

```
DEFINE name value
```

where name is a single word, and value is the rest of the line.

After that definition, whenever name is found in the source file, it will be replaced by value. So, if you are going to use the browser (see Chapter 33), it might be convenient to define this:

```
DEFINE ^Choose3 FTPR 2 72
```

so that you can simply insert ^Choose3 when you want to call the browser.

#### A.4.2.1 Unnamed Local Variable Binding

There is a structure that allows you to refer to local variables with names in the source, but that produces unnamed local variables, thus combining ease of use with speed.

```
The local variables are bound with
```

```
\{\{ \text{ name1 name2 } \dots \text{ nameN } \}\}
```

After that, entering name1 will become 1GETLAM, name2 will become 2GETLAM. Preceding the name of a variable with = or ! stores something in the variable, that is, =name1 becomes 1PUTLAM, and so on.

Pay attention to the way the names are bound: the first variable name corresponds to 1GETLAM (that is, the object that was in level one), the second to 2GETLAM (the object that was in level two), and so on. *This is the opposite of what JAZZ does*. It is possible, however, to get the ordering as JAZZ does, by putting the !JAZZ directive in the beginning of the source file.

#### A.4.2.2 Including Source Files

Using the INCLUDE pseudo-command, you can include other source files in your main program. This is like the #include directive in C programs.

It is used like this: INCLUDE *variable*. The contents of the named variable are read as if they were included in the source file. The included file should also end with an "@".

One use of this feature is to include a file with definitions of several constants or unsupported addresses.

### A.5 Disassembly

As it was briefly mentioned in the description of Library 256 (see section A.3), the command  $\rightarrow$ S2 is the System RPL disassembler. It will disassemble any object in level one into its source code suitable for reassembly with MASD. Unfortunately, there are still some bugs in MASD, which prevent some disassembled objects to be correctly re-assembled. We all hope that in a newer version this bugs will be corrected.

Often, one wants to see how one of the built-in commands in the HP's ROM is built. The JAZZ library for the HP48 calculators made that easy. Unfortunately, it is difficult to do that with only the built-in tools in the HP49G. There are, however, two two libraries for this purpose: Nosy by Jurjen N. E. Boss, and CQIF, by Pierre Tardy. Both allow to extract and disassemble ROM code, and both can be used together with Emacs (see section A.6).

### A.5.1 Using Nosy

Nosy, written by Jurjen N. E. Boss (j.bos@interpay-iss.demon.nl) and presently at version 4.0 is a tool to disassemble the HP49G's ROM. It is very easy to use, and, unlike CQIF?, can be easily used wihout Emacs. It also displays more information that CQIF?, such as the names of flash pointers. Because of this, it is slower than CQIF?

To use Nosy, put the entry name, pointer, an address in the stack (some other inputs are also accepted — see Nosy's documentation) and run the command Nosy. This will open an interactive browser where you can view the disassembled entry and browse the ROM like a hypertext document. Use the arrow keys to scroll. You can quickly view another command inside the disas-

sembled source by moving the highlight to it and pressing ENTER or F6. This will open another browser just like the first one. To go back one level, press Backspace, and to exit press ON.

There are some other functions you can use in the interactive browser, consult the documentation for details.

### A.5.2 Using CQIF?

With the help of the CQIF? (*Comment Qu'Ils Font*?) library, written by Pierre Tardy (e-mail: tardyp@iname.com), currently at version 1.7.7F, the task of disassembling built-in commands is also simplified. It contains several tools for the HP49G hacker. We will not describe everything from the library here, read its documentation if you want to know what else it can do for you.

The most useful command is CQIF?. This command is the basic way to disassemble some part of the HP's ROM. It accepts several kinds of inputs. If you give a string with the name of an entry, that entry is disassembled. You can put an address (a hxs), and run CQIF? to disassemble whatever is at that address. It will also accept the entry pointer itself, rompointers and flashpointers. To ease the disassembly of User RPL commands, you can enter the command inside a list or program (that is, enter { DUP } or « DUP » to disassemble the User RPL command DUP).

CQIF? disassembles step-by-step, so if the command is only a pointer to another command, you need run the CQIF? command several times to get to the real code of the command. Just remove any unnecessary junk from the stack, keeping the last result of CQIF? and run it again. Eventually you will reach the command.

Another useful command in the library is DISPATCH. It does a virtual dispatch based on the object types. To use it, put the objects you would use as arguments to some command in the stack. Then, recall that command (probably using CQIF?) to level one. Run DISPATCH. The object that would be run for those argument types (by means of some dispatching command like CKn&Dispatch) is put in level one.

The other commands are not so useful to System RPL programs. But it is a nice idea to read the documentation and see what CQIF? can do for you.

### A.6 The Editor, and Emacs

When you use the HP49G to develop programs, you will spend most of the time writing or changing the source code. This is done in the editor.

The HP49G editor is much better than the one in the HP48 calculators. However, it can be made even better. There are two variables that are run before entering and after leaving the editor. We will see what can be done with them. We will also describe a library that enhances the editor with some features useful in particular for programming.

Before starting the editor, the variable STARTED is evaluated. You can put a program in this variable to be run before editing any object. And, after leaving the editor, the EXITED variable is run. There are many things these variables can do. A very simple (and very useful) thing is to remove the header during editing, giving a few more lines of text. After the editor is exited, the header is restored to the default setting. It is very simple to do this: STARTED just needs to clear the header:

«  $0 \rightarrow \text{HEADER}$  »

And EXITED restores the header:

« 2 →HEADER »

Change 2 to 1 if you normally use only one line of header. Note that Emacs (see below) removes the header automatically.

For even better customization of the editor there is the Emacs library, written by one of us (CD, e-mail: dominik@astro.uva.nl) and Peter Geelhoed (e-mail: P.F.Geelhoed@student.tnw.tudelft.nl). This library gives the editor some of the features of the famous GNU Emacs editor, such as completion, automatic indentation, incremental search, regular expression search and a macro language. The latest version, at the time of this writing, is 1.10. Again, we will not describe everything in the library — see the manual for more information.

Probably the single most useful feature of the Emacs library is command completion. It is activated by the RPLCPL command. This is only useful in edit mode, so you will need it assigned to a key, with TakeOver before. If you have the KEYMAN library (see section A.2), just put a program like this in the stack:

```
« RPLCPL »
```

and run  $\rightarrow$ TO?. Then, assign the resulting object to a key. It is a nice idea to assign it to the same key both with and without the alpha-mode on. Of course, you do not need a program. Just the rompointer (got with { RPLCPL } HEAD or some similar trick) is enough, but you must still run  $\rightarrow$ TO?.

To try it, enter the first few letters of any User RPL command. Press the key to which you assigned RPLCPL. If there was only one command starting with those letters, what you typed will be completed. If there were more than one, a choose box will appear from which you can select the derised command. The command line will be completed. This is something *really* useful.

Provided you have the extable library installed (as you should — see section A.1), the completion also works for System RPL command names. If the last character in the string is a @ (as required by MASD), then System RPL completion is automatically used. As an added bonus, if you press the key to which RPLCPL is assigned longer, then the lookup of System RPL commands is done with GETNEAR (see section A.1). You can then enter case, ask for completion, and get all words that have case in the middle — not only in the beginning.

Another command that sometimes is useful is DYNCPL. It should also be assigned to some key, and also does completion. But it looks in the file you are editing for words that start with the typed letters. It is useful for the names of local variables and such. It works in a slightly different way: press the key, and the word will be completed with the first word. To accept it, press ENTER. To abort, press ON. To search for another match, press the same key that invoked DYNCPL. Any other key will accept the match and execute that key.

One more command that is useful to be assigned to a key is RPLED. This command imitates the ED command in the JAZZ library: it decompiles the object in level one, opens an editor for you to edit it, and, upon exit, recompiles the object (if you are lucky, that is. If the object cannot be compiled because of some MASD bug, exit the editor with a longer-pressed ENTER. This will allow you to select not to compile the file).

RPLED also displays a menu with useful operations, described below. If you call RPLED when in edit mode, the menu is redisplayed.

To get a description of all the commands in the menu, read the documentation that comes with Emacs. Here we will present the most useful ones:

 ${\tt CO...}$  calls  ${\tt RPLCPL}.$  Left-shift  ${\tt CO...}$  calls  ${\tt DYNCPL}.$  See above for explanations of these commands.

With  $\mid$  > you can collect a few keystrokes into a macro and then run this macro over and over. Press left-shift  $\mid$  > to start the macro recorder, then execute the commands which should be part of the macro and exit with ON. Then use  $\mid$  > to run the macro. Holding down the  $\mid$  > key automatically repeats the macro until you release the key.

Find starts an incremental search. Press this key, then start typing

the string you want to find. Type as many characters as necessary, then press ENTER to go to that cursor position. To cancel the search and go back to where the search started, press ON. Press the right arrow to find the next match.

When you press Left-shift Find, you start a (non-incremental) regular expression search. Read Emacs' manual for more information.

Meta starts a special mode in which many useful editing commands are directly accessible with single key presses. The transmit indication is on while this mode is active. To exit, press ENTER or ON.

Left-shift Meta suspends the editor and goes back to the stack. To return to the editor, press CONT (Left-shift ON).

Right-shift Help is the menu of Emacs configuration. A choose box appears with several actions. Selecting Options will show a dialog, which allow you to configure some aspects of Emacs: whether the minifont is used by default, whether the third page of the menu contains some templates for System RPL and Assembly Language development, the library to use by the EDOB command (described below), and some other things. There are also options to edit the emacs variable (which allows one to add macro commands to Emacs. Again, we refer you to Emacs' documentation), to edit the diagram variable (used by the SDiag library, but this variable is not discussed in this document), and to make some key assignments.

Pressing Left-shift Help toggles between the minifont and the current font.

Indnt indents the current line according to context. However, it is better to write the code already indented than to correct it later... Still, sometimes (such as when cutting and pasting), this can save some time. When left-shifted, removes \*\_ from the beginning of the current line (or all the selected lines), and when pressed right-shifted inserts the \*\_..

- $\{\leftrightarrow\}$ , when pressed in a delimiter, jumps to the matching one. Works with :: and i,  $\{$  and  $\}$  and a few others pairs.
- (  $\rightarrow$  ) shows the stack diagram for the entry point under the cursor. For this to work, the SDiag library, distributed with Emacs, must the installed. All stack diagrams listed in this book are also available on the calculator through this library, and this can be of great help.

EDOB can be used to look into the ROM and into the contents of variables without exiting the editor. If you have ever used JAZZ's ED editor, this works similarly to the Right-shift Y key. It disassembles the entry under the cursor, and its source is viewed in another editor. Exit this sub-editor with ON or ENTER to go back to the original editing section. Of course, you can call DOB

again in the sub-editor. This command requires the CQIF? or Nosy libraries (see section A.5). When Nosy is used, you can press this key longer to run the Nosy browsing environment. (The default is to start a new sub-editor.)

You should assign EDOB to a key since you will use it frequently. Because of the similarity with the Nosy and CQIF? commands, is suitable to assign both commands to the same key. When used in edit mode, EDOB is called. When not, Nosy or CQIF? is called. It is very easy to create an assignment like this with KEYMAN (see section A.2). First, put the list { EDOB Nosy } in the stack, and use OBJ  $\rightarrow$  or COMP  $\rightarrow$  to explode it. Drop the number of objects and run IfE?P. Use  $\rightarrow$ TO? to add TakeOver to the object (since it needs to work in edit mode), and assign it to a key. If you have used the HP48 and JAZZ, Rightshift-hold +/- or Right-shift-hold 1/x will remind ED, and will not interfere with the normal operation. You can replace Nosy with CQIF? here if you prefer the latter library.

Actually, while the above example is very educational, it is not really necessary. EDOB automatically calls CQIF? or Nosy (depending on Emacs' settings) when it is called outside edit mode.

The last page of the Emacs menu contains some templates for System RPL and Assembly Language programming. Try them, you will easily discover what they do.

If you need help with Emacs menu commands, just press Help. It has help on the commands and on the menu keys. If you select help on the meny keys, it will display a screen describing the two pages of the Emacs menu. Each page is represented by three rows of labels, which mean, from top to bottom, the unshifted action, the left-shifted action and the right-shifted action. Some commands are inverted, these have different actions when pressed longer.

There is much more that Emacs can do. Please read the documentation to discover about the rest of the features.

### A.7 Debugging

The debugging facilities for System RPL of the HP49G are the same as for User RPL: the built-in debugger (Left-shift CAT, NXT twice and  ${\tt RUN}$ ). Unfortunately, it does not work very well with some commands, which will be described later.

To start debugging, put the program or the name of the variable in which the program is stored in level one and press DBUG. Then, use the other

commands to examine the program. The SST command executes the next step in the program and displays what has just been executed. You will need the System RPL stack (see section A.3) active for this to be useful. If the command being run is a sub-routine, SST executes this as a single step. SST $\downarrow$  is similar, but if the command is a sub-routine, it steps into this sub-routine and executes its first command.

To see the next two actions of the program, but not execute them, press NEXT. To stop the program being debugged, press KILL. To make it resume its normal operation, press CONT (Left-shift ON).

To insert a breakpoint into your program, insert the command HALT (xhalt for System RPL programmers) in the program at the point you want the program to stop. Then use the commands above to debug the program.

The debugger does not work with commands that take arguments from the runstream, such as ' or IT. Do not try stepping over one of these commands, the only thing you will get is a nice crash :-). Currently, the only way to debug these commands is by inserting xHALT after these commands, and using CONT to skip past the next xHALT.

For simple to moderately complex programs, the procedures described above will be sufficient to find and correct bugs. If you run into a more serious problem with a complicated program, a bigger hammer may be needed: SDB in Jazz49.

### A.8 JAZZ for the HP49

The JAZZ library, written originally by Mika Hesikanen and others for the HP48, implemented many of the features so far discussed in this chapter in a single, compact and very consistent library. On the HP48, this was without any doubt the best programming environment. JAZZ has been ported to the HP49 by Daniel Lidström (e-mail: danli97@ite.mh.se). However, at the time of this writing it is not a full replacement for MASD and the other tools. In particular it has no support for flashpointers and therefore cannot assemble or disassemble programs containing flashpointers. Another drawback is that Jazz needs to be installed in port 0, occupying 70kB (50kB for the light version) of RAM space. It also needs its own table of entry points, 40kb more, but that can fortunately be installed in any port.

The area where Jazz49 brings unique functionality to the HP49 is debugging, both of machine language programs (DB) and of System RPL programs

(SDB). In contrast to the HP49 built-in debugger, SDB can handle runstream commands correctly, so there is no need to insert many xhalt commands into the program. The lack of flashpointer support means that you cannot single-step the contents of flashpointers. Also the display of current and next commands in the status line is affected by this: when the program is near a flash-pointer, "Invalid Object" will be displayed instead of the current and next commands.

To use the debugger, put the program to be debugged or just its name in level 1, and run the SDB command. You will then be presented a menu with your possible actions.  $\rightarrow$ SST executes the next step.  $\rightarrow$ IN is similar, but it will step inside of sub-routines. Use SNXT to show the next steps to be executed. You can insert breakpoints into programs with the SHALT command (write xSHALT in System RPL programs). Note that SHALT only works if SDB is already running, so you need to start your program with SDB and then press CONT to jump to the break point.

Very useful is also the possibility to browse loop and LAM environments with the LOOPS and LAMS commands, respectively. For more information on these and other commands, please refer to the JAZZ documentation.

Now you may wonder if you should really sacrifice 50kB of RAM for the occasional need to do serious debugging. Here is a solution: keep a BZ compressed version of the JAZZ library (light version) stored in port 2 under the name "Jazz". When you need to debug, you can quickly install JAZZ with a small programm:

```
\ll :2: Jazz RCL \sim 0. STO 992. ATTACH \gg
```

where  $\tilde{\ }$  is the decompressor program in OT49. With a similar program, you remove it from port 0 when you are done.

## Appendix B Creating Libraries

Libraries are collections of commands that the user can access as if they were built-in in the system. If you have written a complex program with several sub-routines, it is much more convenient to distribute it as a library instead of as a directory. As a library, the user will not need to navigate through the variables to access your program; he can just type the command name from anywhere. Library commands appear in the catalog, and they can have on-line help. There is a menu showing all installed libraries, and a library can add itself or selected commands to some of the menus, such as the APPS menu.

Moreover, you can make only some of the commands in the library accessible to the user. This way, you can prevent the user from running commands that they should not, and you only need to provide error-checking for the user-accessible commands.

That should have been enough to convince you to distribute your programs as libraries. But you might be wondering, "But how do I *create* a library?"

Easy: the CRLIB command in library 256 (see section A.3) will do that for you. You just need to create a few special variables in a directory, which specify some aspects of the library, and then run that command. You will then get a library from the contents of the directory, which can be distributed.

Instead of CRLIB, you can use the  $D \hookrightarrow L$  command from the OT49 library (see section A.3.1). This command eases the entry of some of the variables below, and provides an easy way to add help to library commands. However, it does not add anything really new to the library creation process.

### **B.1** The Special Variables

In the directory that will be converted to a library, some variables, all having names starting with \$, have special meanings that configure the created library. The table below lists the variables and their meanings.

Variable	Meaning
\$ROMID	This specifies the number of the library. Each library should
	have a unique number, since there cannot be two libraries with
	the same number. It should be a real or an integer, in the range
	769 to 1791.
\$TITLE	This is the title of the library. The first five characters will be
	shown in the library menu. You can have a library without a ti-
	tle, but you will not be able to access the library from the library
***** C = D = D	menu.
\$VISIBLE	This is a list of variable names. The variables listed here will
\$HIDDEN	be made into user-accessible commands in the resulting library. This is a list of variable names. The variables listed here will
PHIDDEN	be converted into hidden commands in the resulting library.
\$CONFIG	This is the library configuration object. This object will be eval-
\$COM 10	uated at each warmstart. Normally, these configuration pro-
	grams attach the library. This can be done by storing some-
	thing like ":: romid TOSRRP;" here, where romid is the li-
	brary id. If you want, you can simply store the real number 1.
	in \$CONFIG, and a default configuration object will be produce,
	which attachs the library at each warmstart.
\$MESSAGE	This is a list of strings which will be available in the library for
	use as (error) messages or general strings. If each message is
	only used once, it is not really worthwhile to create a message
	table. But if messages are used in many places, or if you want to
	make it easy to change messages to a different language, a mes-
	sage table is very useful. The list can contain up to 256 strings.
	Each message on the calculator is identified by a unique bint
	#111mm consisting of a 3-digit library number (like 6FE) and a two-digit message number 01FF. To access a message from a
	program use "#111mm JstGetTHEMESG". To throw an error us-
	ing a message number, use "#111mm DO#EXIT". See Chapter 22
	for more information.
\$EXTPRG	This is the name of a command that allows customization of
	some menus, addition of help to commands and more. See below
	for more information on this.

Note that unlike other library creators, only the variables that are listed in  ${\tt \$VISIBLE}$  or  ${\tt \$HIDDEN}$  are made into command in the library. Variables that do not appear in either list are not converted.  ${\tt \$MESSAGE}$  is optional, you do not have to specify it.

### **B.2** The Library Message Handler

Libraries on the HP49G can contain a message handler. This program is called by the operating system at various occasions, in order to give the library a chance to modify menus, provide online help for its commands and other actions.

When creating a library from a directory, the reserved variable \$EXTPRG can contain the name of a variable in the directory which will later become a rompointer in the library. This rompointer must be a program which accepts a bint on level one (a code representing one of the messages) and, depending upon the specific message, other arguments on higher stack levels.

#### **B.2.1 Menu Extensions**

The majority of messages can be used to extend some built-in menus. Among these are the APPS choose menu, several other choose menus, the SEARCH, GOTO and Tools submenus in the editor menu etc. When the message handler is called to extend a menu, the current menu is on the stack either as a list or as a meta. The program can then modify this menu and return it. So, the stack diagram for menu extensions is one of:

```
({ key1 ... keyN } #msg \rightarrow modified_list #msg)
(key1 ... keyN #n #msg \rightarrow modified_meta #msg)
```

The message number bint stays on the stack, so that the message handler of another library can be called immediately to do its work in the same way.

The following menus on the HP49G can be extended using library messages.

#msg	Menu	Menu Type
0	APPS	list
1	Main STAT menu	list
2	Hypothesis submenu in STAT	list
3	Confidence Interval submenu in STAT menu	list
4	Finance menu	list
5	Numeric Solver menu	list
6	Time menu	list
8	Games (inside APPS)	meta
11	Editor SEARCH menu (when flag -117 is clr)	list
12	Editor TOOLS menu (when flag -117 is clr)	list

#msg	Menu	Menu Type
13	Editor GOTO (when flag -117 is clr)	list
14	Editor SEARCH menu (when flag -117 is set)	meta
15	Editor TOOLS menu (when flag -117 is set)	meta
16	Editor GOTO (when flag -117 is set)	meta

As an example, we show a message handler of a library whose ROMID is 1234. This handler will will add the library menu to the APPS menu, and a particular rompointer to the Games menu. When adding to the APPS menu, the example also makes sure that the new item is numbered just like the other items in the APPS menu. This should be done by all libraries.

```
ZERO OVER#=case
                                    (APPS menu)
        SWAPINCOMP
                                    (save #msq, explode list)
                                    (make index for new entry)
5
        #1+DUP #>$
        ".My Library" !append$
                                    (add name to index number)
        ' :: % 1234. InitMenu%;
                                    (action: set my menu)
        TWO{}N
                                    (label & action -> list)
        SWAP P{}N
                                    (add new entry)
10
        SWAP
                                    (get the ZERO back)
      EIGHT OVER#=case
                                    (Games submenu)
      ::
        DROP
                                    (drop the message)
15
        { "PlayMe" ROMPTR 4D2 0 } (new entry for menu)
        SWAP#1+
                                    (add to meta)
        EIGHT
                                    (put msg number back)
```

### **B.2.2** Online Help for Libary Commands

On the HP49G, all the CAS commands have a short help text which can be displayed from the catalog, or with the SDIAG command in the Emacs library (see section A.6). When the catalog choose box highlights a CAS command, the menu under the choose box has an additional button, the HELP button. Pressing this button shows the corresponding help text. External libraries can provide help for their commands in a similar way, using the message handler and messages number nine and ten. Message nine is a query if the library provides help for a given rompointer. The stack diagram is

```
(romptr FALSE NINE \rightarrow romptr TRUE/FALSE NINE)
```

where the TRUE/FALSE in stack level two indicates if the library is prepared to provide help for the rompointer in level three. This message is used to determine if the HELP button in the CATalog should be turned on.

Message ten is then used to actually display the help when the user presses the HELP button. The stack diagram here is

```
(romptr TEN \rightarrow FALSE)
```

Before pushing FALSE, the message handler should display the help text.

The following example is a message handler which provides a short help string for every visible command in the library.

```
1
    ::
      NINE #=casedrop
        DROPTRUE NINE
                                    (all cmds have help)
5
      TEN #=casedrop
        DUP DECOMP$ NEWLINE&$ SWAP (save cmd name as string)
        ROMPTR># SWAPDROP
                                     (index of romptr in lib)
10
                                     (list of help strings)
           "Help text for romptr 0"
          "Help text for romptr 1"
           "Help text for romptr N" (last visible rompointer)
15
        SWAP#1+ NTHCOMPDROP
                                     (extract correct help str)
                                     (Add the command name)
        FALSE SWAP ViewStrObject
                                    (display the text)
20
```

Note that ViewStrObject conveniently pushes FALSE on the stack, which is the required return value of message nine. The message handler gets a bit more complicated if help is only provided for a few rompointers. In this case, the handler of message nine must check the rompointer against a list, and message ten must use Lookup or something similar to extract the help text. Instead of simply displaying a string, message ten can also do more complicated things, like launching a whole application to provide help.

The library creator in the OT49 library (see section A.3.1) provides a simple way to add help support to a library.

### **B.2.3** The Library Menu Message

If the menu of a library is invoked via the LIBS menu (rightshift 2), the romid of this library is sent to the message handler of the library. The library may use this for easter egg-like stuff (displaying an icon (see for example the Libman library), doing something funny with the menu (e.g. LTool) or playing a melody). It can also change the menu settings, for example to provide functionality for the shifted menu buttons (e.g. ConstTools).

```
The stack diagram for this message is (\#romid \rightarrow \#romid)
```

The following example is the message handler of a library # 60F and it temporarily displays a copyright notice when the library menu is selected.

```
1 ::
    # 60F OVER#=case
    ::
    ZEROZERO
5    "(c) 2001 Some Author"
    $>grob XYGROBDISP
    SetDAlTemp
    ;
;
```

## **Appendix C User RPL Commands**

The listing here is of all the user-accessible commands and functions, with their addresses. In most cases, the User RPL name of a command is equal to the System RPL name with leading  $\tilde{\ }$  and x stripped. The few exceptions are marked in the table.

Addr.	Name	Description
030314	~xABCUV	( pa pb c $ ightarrow$ u v )
39A07	xABS	$(x \rightarrow x')$
390E4	xACK	$( \rightarrow )$
390C9	xACKALL	$( \rightarrow )$
025314	~xACOS2S	$(symb \rightarrow symb')$
3A8D8	xACOSH	$(x \rightarrow x')$
3A7DC	xACOS	$(x \rightarrow x')$
06E314	$\sim$ xADDTMOD	( $symb1 \ symb2 \rightarrow symb3$ )
0000DE	xADDTOREAL	$( var \rightarrow )$
3AAE5	xALOG	$(x \rightarrow x')$
04B0AB	xAMORT	( n $ ightarrow$ princ intr bal )
3CA07	xAND	$(x1 x2 \rightarrow x3)$
3F033	xANS	$(n \rightarrow ob)$
3D7AC	xAPPLY	( {symb1 symbn} f $ ightarrow$
		f(symb1symbn))
3EAC7	xARCHIVE	( :port:name $\rightarrow$ )
		$(:IO:name \rightarrow)$
3C8C6	xARC	( c r $ heta$ 1 $ heta$ 2 $ o$ )
		( $\{ \text{#x #y} \} \text{ #r } \theta 1 \theta 2 \rightarrow )$
3A390	xARG	( c $\rightarrow$ $\theta$ )
085314	~xARIT	$( \rightarrow )$

Addr.	Name	Description
3BEC5	xARRY>	$( [] \rightarrow x1xn \{n\} )$
		( [[]] $\rightarrow$ x11xnm $\{n \ m\}$ )
		$\textbf{UserRPL:} \ \mathtt{ARRY} {\rightarrow}$
3BE9B	x>ARRY	$(x1xn n \rightarrow [])$
		( x11xnm $\{n\ m\} \rightarrow [[]]$ )
		$\mathbf{UserRPL} :  o \mathtt{ARRY}$
024314	~xASIN2C	( $symb \rightarrow symb'$ )
023314	~xASIN2T	( $symb \rightarrow symb'$ )
3A88E	xASINH	$(x \rightarrow x')$
3A756	xASIN	$(x \rightarrow x')$
3EEE7	xASN	( obj key $ ightarrow$ )
		( 'SKEY' $\rightarrow$ )
38DE1	xASR	( # -> #' )
022314	~xATAN2S	( $symb \rightarrow symb'$ )
3A94F	xATANH	$(x \rightarrow x')$
3A844	XATAN	$(x \rightarrow x')$
3EB64	xATTACH	$(n \rightarrow )$
3C49F	XAUTO	$( \rightarrow )$
3C3B2	xAXES	$(c \rightarrow )$
		( {c tick \$x \$y } $ ightarrow$ )
04A314	~xAXL	$( \ \{\} \ \rightarrow \ [\ ] \ )$
		$( [] \rightarrow () )$
	~xAXM	$( [A] \rightarrow [M] )$
04C314	~xAXQ	$([nxn] [n] \rightarrow [nxn]' [n])$
3C9D3	xBAR	$( \rightarrow )$
3E196	xBARPLOT	$( \rightarrow )$
080314	~xBASE	$( \rightarrow )$
		aka: xALGB
3EDCC	xBAUD	$(n \rightarrow )$
39765	xBEEP	( freq dur $ ightarrow$ )
	xBESTFIT	$( \rightarrow )$
3B655	xBIN	$( \rightarrow )$
3E171	xBINS	( min width n $ ightarrow$ [[]] [] )
3C70A	xBLANK	( $\#$ width $\#$ height $ o$ grob )
3C6E0	xBOX	( $\{\#n1 \#m1\} \{\#n2 \#m2\} \rightarrow$ )
		( c1 c2 $\rightarrow$ )
38F21	xB>R	$( \# \rightarrow R )$
		UserRPL: $\mathtt{B}{ ightarrow}\mathtt{R}$
3EE47	xBUFLEN	( $\rightarrow$ nchars 0/1 )

Addr.	Name	Description
39480	xBYTES	( obj $ ightarrow$ chksum size )
01E0DE	xC2P	( $\{\} \rightarrow ?????$ )
07E314	~xCASCFG	$( \rightarrow )$
0330DE	xCASCMD	$( \rightarrow ? )$
38B28	xCASE	$( \rightarrow )$
3AD1B	xCEIL	$(x \rightarrow n)$
3C3DC	xCENTR	$((x,y) \rightarrow )$
		$(x \rightarrow )$
3B4E9	xCF	$(n \rightarrow )$
03A314	$\sim$ xCHINREM	$( []1 []2 \rightarrow []3 )$
04D0AB	xCHOOSE	( title {elems} pos $ ightarrow$ ob 1 )
		( title {elems} pos $ ightarrow$ 0 )
3BC19	xCHR	$(n \rightarrow \$)$
3B362	x%CH	( $x1 x2 \rightarrow x3$ )
01D0DE	xCIRC	( prg $\{\}  ightarrow  ext{????}$ )
3EDAC	xCKSM	( $n_{type} \rightarrow$ )
3DD4E	xCLEAR	( obl obn $\rightarrow$ )
3DD8E	xCLSIGMA	$( \rightarrow )$
		UserRPL: $\mathtt{CL}\Sigma$
	xCLKADJ	$($ ticks $\rightarrow$ $)$
	xCLLCD	$( \rightarrow )$
	xCLOSEIO	$( \rightarrow )$
3E91A	xCLUSR	$(\rightarrow)$
		UserRPL: CLVAR
	~xCMPLX	$(\  ightarrow\ )$
3B193		$([] \rightarrow col\_norm)$
	xCOLCT	$(symb \rightarrow symb')$
	xCOLLECT	$(symb \rightarrow symb')$
3E0FD	xSIGMACOL	$(x_{col} y_{col} \rightarrow )$
		UserRPL: $\mathtt{COL}\Sigma$
0380AB	$x{ ightarrow}COL$	$([[]] \rightarrow [v1][vn] n)$
		$([] \rightarrow x1xn n)$
03E0AB	xCOL-	$([] n \rightarrow []' xn)$
		$([[]] n \rightarrow [[]]' [vn])$
0390AB	$xCOL \rightarrow$	$([v1][vn] n \rightarrow [[]])$
025075	GOT :	$(x1xn n \rightarrow [])$
03F0AB	xCOL+	$([[]][[]]' n \rightarrow [[]]'')$
		( [] $x n \rightarrow$ []')

Addr.	Name	Description
3B423	xCOMB	( n k $\rightarrow$ Cn,k )
		Symbolic argument allowed.
0260AB	xCOND	( $[[n*n]] \rightarrow x$ )
3C967	xCONIC	$( \rightarrow )$
39A6C	xCONJ	$(x \rightarrow x')$
0180AB	xCONLIB	$( \rightarrow )$
3BF77	xCON	$( \{ n \} x \rightarrow [] )$
		$( \{ n k \} x \rightarrow [[]] )$
		$( [] X \rightarrow []' )$
0190AB	xCONST	( name $\rightarrow$ x )
3989C	xCONT	$( \rightarrow )$
38F41	xCONVERT	( $x1_u1 x2_u2 \rightarrow x3_u2$ )
3DE24		( $ ightarrow$ x_correlation )
3A6C2	xCOSH	$(x \rightarrow x')$
3A5D0	xCOS	$(x \rightarrow x')$
3DE3F	xCOV	( $ ightarrow$ x_covariance )
3C58E	xC>PX	$((x,y) \rightarrow \{\#n \#m\})$
		UserRPL: $C \rightarrow PX$
393CA	xCRDIR	( name $ ightarrow$ )
3D128	xCR	$( \rightarrow )$
	xCROSS	$([1][2] \rightarrow [3])$
3BAF5	xC>R	$(x,y) \rightarrow x y$
		UserRPL: $C \rightarrow R$
	~xCURL	$([func] [vars] \rightarrow [])$
	xCYLIN	$(\rightarrow)$
	xDARCY	( $xe/D$ $yRe \rightarrow xDarcy$ )
	xSETDATE	( date → )
39078		$(\rightarrow date)$
	xDATE+	( date ndays → date' )
0690AB		( → %1 )
0150DD	xDBUG	( prog → )
		$(name \rightarrow )$
39218	xDDAYS	( date1 date2 $\rightarrow$ days )
3B670	xDEC	$(\rightarrow)$
3E576	xDECR	$( name \rightarrow x_new )$
3E85C	xDEFINE	$( 'name=expr' \rightarrow )$
		( 'name(name1) = expr(name1)
3DE 40	DEG	$\rightarrow$ )
3B549	xDEG	$(\  ightarrow\ )$

Addr.	Name	Description
391D8	xDELALARM	$(n \rightarrow )$
3D1C7	xDELAY	( $x_{delay} \rightarrow$ )
3EF3B	xDELKEYS	( rc.p $\rightarrow$ )
		$(0 \rightarrow )$
		$( \ \ 'S' \  o \ )$
3C51F	xDEPND	( name $ ightarrow$ )
		( $\{ \text{name y1 y2} \} \rightarrow )$
		( $\{y1\ y2\} \rightarrow$ )
		( y1 y2 $\rightarrow$ )
3DCA7	xDEPTH	$( \rightarrow n )$
00E314	~xDERIV	( $symb \ var \rightarrow symb'$ )
003314	~xDERVX	( $symb \rightarrow symb'$ )
00F314	$\sim$ xDESOLVE	( eq func $ ightarrow$ func' )
3B1BA	xDET	$( [[]] \rightarrow x )$
3EB84	xDETACH	$(n \rightarrow )$
		$(:port:n \rightarrow)$
3D202	$x\partial$	( $symb \ var \rightarrow symb'$ )
03A0AB	$x \rightarrow DIAG$	$([[]] \rightarrow \text{vec})$
03B0AB	$\texttt{xDIAG} {\rightarrow}$	( [] $\{$ dims $\}$ $ ightarrow$ [[]] )
084314	~xDIFF	$( \rightarrow )$
00E0AB	~	$( \rightarrow )$
39725		( obj n_line $\rightarrow$ )
0160DD	xDISPXY	( ob $\{ \text{#x #y} \} \text{ %size } \rightarrow \text{ )}$
		Display ob (decompiled if nexessary) at
		the given display coordinates, using either
		the system font (%size=2) or the minifont
		(%size=1).
056314		([func] [vars] → func)
	~xDIV2	$( \text{symb1 symb2} \rightarrow \text{squot srem})$
	~xDIV2MOD	$( \text{symb1 symb2} \rightarrow \text{squot srem})$
	~xDIVIS	$(symb \rightarrow \{\})$
071314		$( \text{symb1 symb2} \rightarrow \text{sq} )$
	~xDIVPC	( symb1 symb2 n $\rightarrow$ symb3 )
3816B	xD0	$( \rightarrow )$
39527	xDOERR	$(n \rightarrow )$
		$( \ \ \ \ \ \ \ \ \ \ \ )$
055035	DOI T.O.	$(0 \rightarrow )$
OPROAR	xDOLIST	$(\{1\},\ldots\{n\} \text{ n prog } \rightarrow \{\})$
		( $\{1\}\dots\{n\}$ prog $\rightarrow$ $\{\}$ (n=1) )

Addr.	Name	Description
0540AB	xDOSUBS	( {} n prog → {}')
		( $\{\} \text{ prog } \rightarrow \{\} \text{ ' (n=1) )}$
3B1E1	xDOT	$([1][2] \rightarrow x)$
3C484	xDRAW	$( \rightarrow )$
06B0AB	xDRAW3DMATRIX	( [[]] $v_{min} v_{max} \rightarrow$ )
3C4BA	xDRAX	$( \rightarrow )$
3DC56	xDROP2	( ob1 ob2 $ ightarrow$ )
3DCC7	xDROPN	$(ob1obn n \rightarrow )$
3DC3B	xDROP	( ob $\rightarrow$ )
3B06E	xD>R	( $x \rightarrow (\pi/180)x$ )
		UserRPL: $D \rightarrow R$
3EFEF	xDTAG	( tag:obj → obj )
3DC05	xDUP2	$(12 \rightarrow 1212)$
3F29A	xDUPDUP	( 1 $\rightarrow$ 1 1 )
3DCE2	xDUPN	$(1n n \rightarrow 1n 1n)$
3DBEA	xDUP	( ob $ ightarrow$ ob ob )
0090DD	xEDITB	( ob $\rightarrow$ ob' )
0070DD	xEDIT	( ob $\rightarrow$ ob' )
39B1E	xCONSTANTe	$( \rightarrow e )$
		UserRPL: e
02E314	~xEGCD	( $symb1 symb2 \rightarrow symb3 symb4 symb5 )$
02C0AB	xEGV	$( [[]] \rightarrow [[evect]]' [evals] )$
02D0AB	xEGVL	$([[]] \rightarrow [egval])$
3805D	xELSE	$( \rightarrow )$
38A54	xENDDO	$(1/0 \rightarrow )$
		UserRPL: END
0570AB	xENDSUB	$( \rightarrow x )$
		Number of lists in DOSUBS.
3B5DA	xENG	$(n \rightarrow )$
088314	~xEPSX0	( $symb1 \rightarrow symb2$ )
3BDE6	xEQ>	$( 'l=r' \rightarrow l r )$
		UserRPL: $EQ \rightarrow$
00B0DD	xEQW	$( \text{symb} \rightarrow \text{symb'} )$
3C553	xERASE	$( \rightarrow )$
3955B	xERR0	$( \rightarrow )$
39591	xERRM	( $ ightarrow$ \$msg )
39576	xERRN	( $\rightarrow$ \$nerr )
038314	~xEULER	( $z1 \rightarrow z2$ )
395AC	xEVAL	$(ob \rightarrow ?)$

```
Addr.
           Name
                                 Description
06C314
                                 (symb \rightarrow symb1 symb2)
           ^{\sim}xEXLR
                                 ( symb1 \rightarrow symb2 )
           ^{\sim}xEXPANDMOD
076314
000314
                                 ( symb1 \rightarrow symb2 )
           ~xEXPAND
                                 ([symb1] \rightarrow [symb2])
                                 ( symb1 \rightarrow symb2 )
3E5E9
           XEXPAN
3E25E
           XEXPFIT
                                 ( \rightarrow )
                                 ( symb1 \rightarrow symb2 )
017314
           ~xEXPLN
3AB6F
                                 (x \rightarrow x')
           xEXPM
3A9B7
                                 (x \rightarrow x')
           xEXP
0050AB XEYEPT
                                 (xx xy xz \rightarrow)
0620AB
           xF0\lambda
                                 ( y_{\text{lambda}} \times T \rightarrow x_{\text{power}} )
001314
           ~xFACTOR
                                 (symb \rightarrow symb1*symb2...)
                                 (z \rightarrow z1*z2...)
                                 ( symb \rightarrow symb1*symb2... )
077314
           ~xFACTORMOD
                                 (z \rightarrow \{z1 m1...\})
043314
           ~xFACTORS
                                 (symb \rightarrow \{symb1 m1...\})
                                 ( x_x/D y_Re \rightarrow x_fanning )
0600AB
           xFANNING
                                 ( \rightarrow )
3F2DF
           xFAST3D
3B635
           xFC?C
                                 (n \rightarrow 0/1)
                                 ( n \rightarrow 0/1 )
3B529
           xFC?
           ~xFCOEF
                                 ( [] \rightarrow \text{symb} )
041314
01A0AB xFFT
                                 ( [] \rightarrow []' )
00C0DD xFILER
                                 ( \rightarrow )
391AE
           xFINDALARM
                                 ( date \rightarrow n )
                                 ( {date time} \rightarrow n )
                                 (0 \rightarrow n)
                                 ( \rightarrow )
3ED76
           xFINISH
                                 (n \rightarrow )
3B59A
           xFIX
0170AB xFLASHEVAL
                                 ( \# \rightarrow ? )
                                 (x \rightarrow n)
3ACD1
           xFLOOR
                                 ( \rightarrow font )
00F0DD xFONT6
                                 ( \rightarrow font )
00E0DD xFONT7
                                 ( \rightarrow font )
00D0DD
           xFONT8
                                 ( \rightarrow font )
0030DD
           xFONT \rightarrow
           x \rightarrow FONT
0020DD
                                 (font \rightarrow)
                                 ( start finish \rightarrow )
38252
           xSTARTVAR
                                 UserRPL: FOR
                                 ( symb z \rightarrow c_z )
05E314
           ~xFOURIER
                                 (x \rightarrow x')
3AC87
           xFP
```

Addr.	Name	Description
39745	xFREEZE	$(n \rightarrow )$
042314	~xFROOTS	( symb $ ightarrow$ [])
3B615	xFS?C	( $n \rightarrow 0/1$ )
3B509	xFS?	( n $\rightarrow$ 0/1 )
3C955	xFUNCTION	$(\  ightarrow\ )$
3D56B	x	( symb $\{  ext{var val } \ldots \}  o  ext{x'}$ )
06B314	~xFXND	$( 'x/y' \rightarrow x y )$
0070DE	xGAMMA	$(x \rightarrow x')$
04D314	~xGAUSS	( symb [vars] $ ightarrow$ [diag] [P] symb'
		[vars] )
075314	~xGCDMOD	$(x1 x2 \rightarrow x3)$
02C314	~xGCD	$(x1 x2 \rightarrow x3)$
0550AB	$\mathtt{x}\Delta\mathtt{LIST}$	$( \ \{\} \ \rightarrow \ \{\}' \ )$
3C22D	xGETI	( ob pos $ ightarrow$ ob' pos' elm )
		ob = [] or [[]] or {} or name
		$pos = n or \{n\} or \{n m\}$
3C1C7	xGET	( ob n $ ightarrow$ elm )
		ob = [] or [[]] or {} or name
		$pos = n or \{n\} or \{n m\}$
3C74A	xGOR	( g_targ $\{\#n \ \#m\}$ grob $ o$ g_targ' )
		( $g_{targ}(x,y) grob \rightarrow g_{targ}'$ )
		( PICT $\ldots$ $\rightarrow$ )
3B57F	xGRAD	$( \rightarrow )$
3C5AE	xGRAPH	$( \rightarrow )$
		UserRPL: PICTURE
00A0AB	xGRIDMAP	$( \rightarrow )$
07C314	~xGROBADD	( gr1 gr2 $\rightarrow$ gr3 )
3C8A1	x>GROB	( ob n_chrsize $ ightarrow$ grob )
		UserRPL:  o GROB
3D503	xSUM	( var n1 n2 symb $\rightarrow$ x )
		UserRPL: $\Sigma$
3DDEE	xSIGMA-	$( \rightarrow x )$
		$( \rightarrow [] )$
		UserRPL: $\Sigma$ -
3DDC4	xSIGMA+	$(x \rightarrow )$
		$(x1xn \rightarrow )$
		UserRPL: $\Sigma$ +
3E156	xSIGMALINE	$(\rightarrow \text{symb})$
		UserRPL: $\Sigma$ LINE

Addr.	Name	Description
0590AB	$x\Sigma$ LIST	( {} → x )
3DE90	xSUMX2	( $ ightarrow$ xsum )
		UserRPL: $\Sigma X2$
3DE5A	xSUMX	( $ ightarrow$ xsum )
		UserRPL: $\Sigma X$
3DEC6	xSUMXY	( $ ightarrow$ xsum )
		UserRPL: $\Sigma XY$
3DEAB	xSUMY2	( $ ightarrow$ xsum )
		UserRPL: $\Sigma$ Y2
3DE75	xSUMY	( $ ightarrow$ xsum )
		UserRPL: $\Sigma Y$
3C7D8	xGXOR	( g_targ $\{\#n \ \#m\}$ g_src $ ightarrow$ g_targ' )
		( g_targ (x,y) g_src $\rightarrow$ g_targ' )
		( PICT $\ldots$ $\rightarrow$ )
046314	~xHADAMARD	( [M1] [M2] $\rightarrow$ [M3] )
020314	$^{\sim}$ xHALFTAN	$( symb \rightarrow symb' )$
3880D	xHALT	$( \rightarrow )$
0040DD	$x{ ightarrow} HEADER$	$(n \rightarrow )$
0050DD	$\texttt{xHEADER}{\rightarrow}$	$( \rightarrow n )$
05C314	$\sim$ xHERMITE	( $z \rightarrow symb$ )
059314	~xHESS	(symb [vars] $\rightarrow$ [M] [grad] [vars])
3B68B	xHEX	$( \rightarrow )$
054314	$^{\sim}$ xHILBERT	$(z \rightarrow [M])$
3C9C1	xHISTOGRAM	$( \rightarrow )$
3E1CA	xHISTPLOT	$( \rightarrow )$
3B14C	xHMS-	( hms1 hms2 $\rightarrow$ hms3 )
3B12C	xHMS+	( hms1 hms2 $\rightarrow$ hms3 )
3B0EC	x>HMS	$(x \rightarrow x')$
		$\mathbf{UserRPL} :  o \mathtt{HMS}$
3B10C	$\mathtt{xHMS}>$	$(x \rightarrow x')$
		$UserRPL: {\tt HMS} {\rightarrow}$
	xHOME	$( \rightarrow )$
	~xHORNER	( $symb1 x \rightarrow symb2 x symb3$ )
031314	~xIABCUV	( n1 n2 n3 $\rightarrow$ n4 n5 )
	xIBERNOULLI	$(n \rightarrow x)$
00B314		$(uv'v \rightarrow uv - u'v)$
03B314	~xICHINREM	$( []1 []2 \rightarrow []3 )$
027314	~xIDIV2	( n1 n2 $ ightarrow$ quot rem )

Addr.	Name	Description
3C02E	xIDN	$(n \rightarrow [[]])$
		$( [[]] \rightarrow [[]]' )$
		( name $\rightarrow$ [[]] )
02F314	~xIEGCD	( n1 n2 $\rightarrow$ c b a )
37F48	xIF	$( \rightarrow )$
387AC	xIFERR	$( \rightarrow )$
01B0AB	xIFFT	$( [] \rightarrow []' )$
396A4	xIFT	( $0/1 \text{ obj } \rightarrow ?$ )
395F3	XIFTE	( $0/1$ objT objF $ ightarrow$ ? )
39B3B	xi	$( \rightarrow i )$
011314	~xILAP	( $symb \rightarrow symb'$ )
3B87E	XIM	$((x,y) \rightarrow y)$
		$( [] \rightarrow []' )$
3E54C	xINCR	( name $ ightarrow$ x' )
3C33E	XINDEP	( name $ ightarrow$ )
		( $\{ \text{name x1 x2} \} \rightarrow )$
		( $\{x1 \ x2\} \rightarrow )$
		( x1 x2 $\rightarrow$ )
08A314	~x∞	$( \rightarrow ' + \infty ' )$
		Infinity
04C0AB	xINFORM	( $flds$ fmt $rst$ $init$ $flds$
		)
		( $\{flds\}\ fmt\ \{rst\}\ \{init\}\  o 0$ )
3EEBD	xINPUT	( $prompt $ \rightarrow $' )$
		( $prompt \{ pecs \} \rightarrow \ \ \ )$
3D434	x∫	( x1 x2 symb var $ ightarrow$ symb' )
3F007	XINT	( f(var) var $x0 \rightarrow F(x0)$ )
004314	~xINTVX	$(f(x) \rightarrow F(x))$
074314	~xINVMOD	$(x \rightarrow x')$
3A32B	XINV	$(x \rightarrow 1/x)$
		$( [[]] \rightarrow [[]]' )$
3AC3D	xIP	$(x \rightarrow n)$
029314	~xIQUOT	( $n1 \ n2 \rightarrow n3$ )
02B314	$^{\sim}$ xIREMAINDER	( $n1 \ n2 \rightarrow n3$ )
3F0B7	xI>R	$(n \rightarrow x)$
		UserRPL: $I \rightarrow R$
3E648	xISOL	( symb var $ ightarrow$ symb' )
03C314	~xISPRIME?	$(n \rightarrow 1)$
		( $n \rightarrow 0$ )

```
Addr.
            Name
                                  Description
050314
                                  ([nxn] \rightarrow minpol chrpol {} {})
            ~xJORDAN
3EE2C
            xKERRM
                                  ( \rightarrow \mathsf{msg})
07B314
                                  ( rc.p \rightarrow ?
            ~xKEYEVAL
                                  (\rightarrow rc 1)
39854
            xKEY
                                  ( \rightarrow 0 )
06C0AB
           x \rightarrow KEYTIME
                                  ( ticks \rightarrow )
                                  ( \rightarrow ticks )
06D0AB xKEYTIME→
3ECE4
                                  ( name \rightarrow )
            xKGET
                                  ( "name" \rightarrow )
                                  ( \{names\} \rightarrow )
                                  ( \{\{\text{old new}\}...\} \rightarrow \}
394F1
           xKILL
                                  ( \rightarrow )
                                  ( \rightarrow )
3C5C9
            xLABEL
                                  ( [2xn] \rightarrow pol )
05D314
            ~xLAGRANGE
0000DD x→LANGUAGE
                                  (n \rightarrow )
0010DD xLANGUAGE \rightarrow
                                  ( \rightarrow n )
                                  (symb[vars] \rightarrow symb')
058314
           ^{\sim}xLAPL
                                  ( symb \rightarrow symb' )
            ^{\sim}xLAP
010314
                                  ( \rightarrow ob1 .. obn )
397E5
           xLAST
                                  UserRPL: LASTARG
3C881
                                  (qrob \rightarrow )
           x>LCD
                                  UserRPL: \rightarrow LCD
3C866
                                  ( \rightarrow \text{grob})
           xLCD>
                                  UserRPL: LCD \rightarrow
                                  ( symb1 symb2 \rightarrow symb3 )
02D314
            \simxLCM
055314
            \simxLCXM
                                  ( n1 n2 prog \rightarrow [])
012314
            ~xLDEC
                                  ( symb1 symb2 \rightarrow symb3 )
                                  (n \rightarrow pol)
05A314
            ~xLEGENDRE
032314
           ~xLGCD
                                  ( \{ \text{symb...} \} \rightarrow \{ \} \text{ gcd } )
0160AB XLIBEVAL
                                  ( \# \rightarrow ? )
                                  ( \rightarrow {title nlib nport ...} )
3EB42
            xLIBS
005314
            ^{\sim}xLIMIT
                                  (func point \rightarrow lim)
3C68C
                                  ((x1,y1)(x2,y2) \rightarrow )
            xLINE
                                  ( \{ \#n1 \ \#m1 \} \ \{ \#n2 \ \#m2 \} \rightarrow )
3E214
                                  ( \rightarrow )
           xLINFIT
                                  ( symb var \rightarrow 0/1 )
0150AB XLININ
052314
            ~xLINSOLVE
                                  ([eqs] [vars] \rightarrow [eqs] \{pp\} sol)
014314 ~xLIN
                                  (symb \rightarrow symb')
```

Addr.	Name	Description
3BAC1	xLIST>	$(\{\} \rightarrow ob1obn n)$
		$\textbf{UserRPL:} \ \mathtt{LIST} {\rightarrow}$
3B7D2	x>LIST	( obl obn n $\rightarrow$ {} )
		$\mathbf{UserRPL:}  o \mathtt{LIST}$
06D314	$\sim$ xLNAME	( $symb \rightarrow [vars]$ )
016314	$\sim$ xLNCOLLECT	$( \text{symb} \rightarrow \text{symb'} )$
3AB2F	xLNP1	$(x \rightarrow x')$
3AA01	xLN	$(x \rightarrow x')$
3E239	xLOGFIT	$( \rightarrow )$
3AA73	xLOG	$(x \rightarrow x')$
0320AB	xLQ	$( \hspace{.1cm} \texttt{[[]]} \hspace{.1cm} \rightarrow \hspace{.1cm} \texttt{[[L]]} \hspace{.1cm} \texttt{[[Q]]} \hspace{.1cm} \texttt{[[P]]} \hspace{.1cm} )$
3DF83	xLR	( $ ightarrow$ Intercept Slope )
02B0AB	xLSQ	$( [B] [[A]] \rightarrow []' )$
		( [[B]] [[A]] $\rightarrow$ [[]]' )
0300AB	xLU	$( \hspace{.1cm} \texttt{[[]]} \hspace{.1cm} \rightarrow \hspace{.1cm} \texttt{[[L]]} \hspace{.1cm} \texttt{[[U]]} \hspace{.1cm} \texttt{[[P]]} \hspace{.1cm} )$
06A314	~xLVAR	( $symb \rightarrow symb [vars]$ )
051314	~xMAD	( [] $ ightarrow$ det inv coeff cpol )
3B02E	xMANT	$(x \rightarrow x')$
066314	~xMAP	( $\{\}$ prog $ ightarrow$ $\{\}$ ' )
3DAD0	xMATCHUP	( symb $\{ ext{spat srepl}\}  ightarrow  ext{symb'}$ 0/1 )
		( $symb \{ spat srepl scond \}  ightarrow symb'$
		0/1 )
		UserRPL: ↑MATCH
3DEE1	xMAXSIGMA	( $ ightarrow$ xmax )
		$( \rightarrow [x1xn] )$
		UserRPL: MAX $\Sigma$
39AE4	xMAXR	( $\rightarrow$ MAXR )
3ADA5	XMAX	$(xy \rightarrow x')$
0760AB	xMCALC	$( var \rightarrow )$
		( $\{vars\} \rightarrow )$
		$($ "ALL" $\rightarrow$ $)$
3DEFC	xMEAN	( $ ightarrow$ xmean )
		$( \rightarrow [x1xn] )$
3E8C1	XMEM	$( \rightarrow x )$
3E9D4	xMENU	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
	~xMENUXY	( n1 n2 $\rightarrow$ )
3DF17	xMINSIGMA	$( \rightarrow xmin )$
		$(\rightarrow [x1xn])$
		UserRPL: $\texttt{MIN}\Sigma$

```
Addr.
            Name
                                   Description
0110DD
                                   (font \rightarrow)
            x \rightarrow MINIFONT
                                   ( \rightarrow font )
0120DD
            xMINIFONT \rightarrow
                                   ( \rightarrow )
0730AB
            xMINIT
                                   ( \rightarrow MINR )
39B01
            xMINR
3AE2B
            xMIN
                                   (xy \rightarrow x')
0740AB
            xMITM
                                   ( title \{vars\} \rightarrow )
079314
            \simxMODSTO
                                   (\mod \rightarrow)
                                   (xy \rightarrow x')
3AFCB
            xMOD
0770AB
                                   ( var \rightarrow x )
            xMROOT
                                   ( "ALL" \rightarrow )
                                   ( \$ \rightarrow )
04E0AB
            xMSGBOX
                                   ( \rightarrow )
0720AB
            xMSOLVR
070314
            ~xMULTMOD
                                   ( symb1 symb2 \rightarrow symb3 )
                                   ( var \rightarrow )
0750AB
            xMUSER
                                   ( \{ vars \} \rightarrow )
                                   ( "ALL" \rightarrow )
                                   (n \rightarrow )
0060DD
            x \rightarrow NDISP
                                   ( xq v x \rightarrow x' )
01C0AB
            xNDIST
3F2B5
            xNDUPN
                                   ( ob n \rightarrow ob ..
                                                             ob n )
                                   (x \rightarrow x')
39976
            xNEG
                                   (ob \rightarrow ob)
394AA
            XNEWOB
3831C
            XNEXT
                                   ( \rightarrow )
                                   (n \rightarrow n')
            ~xNEXTPRIME
03D314
3DE09
            xNSIGMA
                                   (\rightarrow \text{nrows})
                                   UserRPL: N\Sigma
                                   ( ob1 ob2 \rightarrow ob2 )
3F264
            xNIP
                                   (x \rightarrow x')
3CB13
            xNOT
                                   ( \rightarrow )
3F0FC
            xNOVAL
0560AB
            xNSUB
                                   ( \rightarrow npos )
                                   ( \$ \rightarrow n )
3BBF9
            xNUM
                                   (x \rightarrow x')
39785
            x>NUM
                                   UserRPL: \rightarrow NUM
                                   (n \rightarrow )
0060AB
            xNUMX
                                   ( n \rightarrow )
0070AB
            \timesNUMY
3885C
                                   ( obl .. obn \rightarrow )
            xRPN->
                                   UserRPL: \rightarrow
3BE38
            xOBJ>
                                   ( ob \rightarrow ?
                                   UserRPL: OBJ \rightarrow
                                   ( \rightarrow )
3B6A6
            xOCT
```

Addr.	Name	Description
3950C	xOFF	$(\  ightarrow\ )$
3EC75	XOPENIO	$( \rightarrow )$
3E8F0	xORDER	( $\{\mathtt{names}\} \rightarrow )$
3CA8D	xOR	$(xy \rightarrow x')$
3DC8C	XOVER	( 1 2 $\rightarrow$ 1 2 1 )
039314	~xPA2B2	$(n \rightarrow n')$
3C98B	xPARAMETRIC	$( \rightarrow )$
3EDEC	xPARITY	$(n \rightarrow )$
0090AB	xPARSURFACE	$( \rightarrow )$
034314	~xPARTFRAC	( $symb \rightarrow symb'$ )
393EA	xPATH	( $ ightarrow$ {HOME dirl $\dots$ dirn} )
04F314	~xPCAR	$( [nxn] \rightarrow pol )$
0450AB	xPCOEF	( [roots] $ ightarrow$ [coefs] )
00D0AB	xPCONTOUR	$( \rightarrow )$
01F0AB	xPCOV	( $ ightarrow$ xpcovariance )
3C4F5	xPDIM	$($ (xmin,ymin) (xmax,ymax) $\rightarrow$ $)$
		( #width #height $ ightarrow$ )
3B477	xPERM	( $n k \rightarrow n'$ )
0460AB	xPEVAL	( [coefs] $x \rightarrow x'$ )
3EAA7	xPGDIR	( name $ ightarrow$ )
3F27F	xPICK3	$(123 \rightarrow 1231)$
3DCFD	xPICK	$(1n n \rightarrow 1n 1)$
3C72A	xPICT	( $ ightarrow$ PICT )
05A0AB	$x\Pi$ LIST	$(\{\}\rightarrow x)$
06A0AB	xPINIT	$( \rightarrow )$
39AC7	xPI	( $\rightarrow \pi$ )
		UserRPL: $\pi$
3C638	xPIXOFF	$((x,y) \rightarrow )$
		$( \{ \#n \ \#m \} \rightarrow )$
3C60E	xPIXON	$((x,y) \rightarrow )$
		$(\{\#n \#m\} \rightarrow)$
3C662	xPIX?	$((x,y) \rightarrow 1/0)$
		$( \{ \#n \ \#m \} \rightarrow 1/0 )$
3EE9D	xPKT	( data type $\rightarrow$ response )
00A314	~xPLOTADD	$(f \rightarrow)$
3C392	xPMAX	$((x,y) \rightarrow )$
3C372	xPMIN	$((x,y) \rightarrow )$
3C979	xPOLAR	$( \rightarrow )$

Addr.	Name	Description
3BB94	xPOS	( str substring $\rightarrow$ n/0 )
		( $\{\}$ ob $\rightarrow$ n/0 )
073314	~xPOWMOD	( symb exp $ ightarrow$ symb' )
3D0D7	xPR1	$( ob \rightarrow ob )$
3DFDD	xPREDV	$(x \rightarrow y)$
3E01D	xPREDX	$(y \rightarrow x)$
3DFFD	xPREDY	$(x \rightarrow y)$
00C314	$^{\sim}$ xPREVAL	( f x1 x2 $\rightarrow$ symb )
		( f x1 x2 $\rightarrow$ x )
03E314	$^{\sim}$ xPREVPRIME	$(n \rightarrow n')$
3D1E7	xPRLCD	$( \rightarrow )$
38BBF	xPROMPT	$( \ \ \ \ \ \ \ \ \ \ \ )$
08B314	~xPROMPTSTO	$($ var $\rightarrow$ $)$
	xPROOT	$([coefs] \rightarrow [roots])$
	~xPROPFRAC	$(x \rightarrow symb')$
	xPRST	$(\  ightarrow\ )$
3D143	xPRVAR	$(\text{name} \rightarrow)$
		( $\{\text{names}\} \rightarrow )$
		$(:port:name \rightarrow)$
01D0AB	xPSDEV	( → xpsdev )
		$(\rightarrow \{x1xn\})$
0040DE		$( symb \rightarrow symb' )$
0030DE		$( \text{symb n} \rightarrow \text{symb'})$
	~xPTAYL	$(pol x \rightarrow pol')$
3E87C	xPURGE	( name → )
		$( \{ \text{names} \} \rightarrow )$
		( :port:name → )
20120	DIMT	(:port:nlib → )
3C139	xPUTI	( ob pos obj $\rightarrow$ [] pos') ob = $[]$ or $[]$ ] or $\{\}$ or name
		$pos = n \text{ or } \{n\} \text{ or } \{n \text{ m}\}$
3C0BF	xPUT	$pos = n \text{ or } \{n\} \text{ or } \{n \text{ in}\}$ $(ob pos obj \rightarrow ob')$
SCUBE	XPUI	$ob = [] or [[]] or {} or name$
		$pos = n \text{ or } \{n\} \text{ or } \{n \text{ m}\}$
3EA49	xPVARS	$\begin{array}{c} pos = H O I I I I I I I I$
3C5E4	xPVIEW	$((x,y) \rightarrow )$
COLT	VE A TRIAA	$( \{ x, y \} \rightarrow ) $ $( \{ \text{#n } \text{#m} \} \rightarrow )$
3C56E	xPX>C	$ \left( \begin{array}{ccc} \{\#\mathbb{I} & \#\mathbb{I} \} & \to & (x,y) \end{array} \right) $
20201	21 27 0	UserRPL: $PX \rightarrow C$
		0.001101 12. 12. 70

Addr.	Name	Description
3DA3E	x->Q	$(x \rightarrow a/b)$
		$\text{UserRPL:} \rightarrow \hspace{-0.5em} \bigcirc$
3DA63	x->QPI	$(x \rightarrow symb)$
		$\textbf{UserRPL:} \rightarrow \hspace{-0.05cm} \bigcirc \pi$
0310AB	xQR	$( \hspace{.1cm} [\hspace{.1cm} [\hspace{.1cm} ]\hspace{.1cm}] \hspace{.1cm} \rightarrow \hspace{.1cm} [\hspace{.1cm} [\hspace{.1cm} Q\hspace{.1cm}] \hspace{.1cm}] \hspace{.1cm} [\hspace{.1cm} [\hspace{.1cm} R\hspace{.1cm}] \hspace{.1cm}] \hspace{.1cm} [\hspace{.1cm} [\hspace{.1cm} P\hspace{.1cm}] \hspace{.1cm}] \hspace{.1cm})$
3E66F	xQUAD	( $symb \ var \  o \ symb'$ )
3D6F6	xQUOTE	$(ob \rightarrow ob)$
028314	$\sim$ xQUOT	( p1 p2 $\rightarrow$ p3 )
04B314	$\sim$ xQXA	( symb [vars] $ ightarrow$ [[]] [vars] )
3B564	xRAD	$( \rightarrow )$
3B3E6	xRAND	$( \rightarrow x )$
02A0AB	xRANK	$( [[]] \rightarrow n )$
0350AB	xRANM	( $\{m \ n\} \rightarrow [[]]$ )
3DBCA	xPREDIV	$(xy \rightarrow x/y)$
		UserRPL: RATIO
38F01	xR>B	$(x \rightarrow \#)$
		UserRPL: $R \rightarrow B$
3D393	xRCEQ	$( \rightarrow EQ )$
3B7ED	xR>C	$(xy \rightarrow (x,y))$
		UserRPL: $R \rightarrow C$
3918E	xRCLALARM	( n $ ightarrow$ {date time action rep} )
3B715	xRCLF	( $\rightarrow$ { $\#$ s1 $\#$ u1 $\#$ s2 $\#$ u2} )
03F0DE	xRCLVX	( $ ightarrow$ name )
		Recall the current content of the reserved
		CAS variable VX.
3DDA9	xRCLSIGMA	$( \rightarrow [[]])$
		UserRPL: $\mathtt{RCL}\Sigma$
3EF79	xRCLKEYS	( $ ightarrow$ {ob key} )
3EA2E	xRCLMENU	$( \rightarrow x )$
3E6F1	xRCL	$( var \rightarrow x )$
		$( : port:nlib \rightarrow lib )$
		$(:port:name \rightarrow ob)$
		$(:port:\{path\} \rightarrow ob)$
3B6FA	xRCWS	$(\rightarrow n)$
3B0AE	xR>D	$(x \rightarrow (180/\pi)x)$
		$\mathbf{U}\mathbf{ser}\mathbf{RPL}$ : $\mathtt{R}{ ightarrow}\mathtt{D}$

Addr.	Name	Description
3BEEC	xRDM	( ob size $\rightarrow$ ob' )
		( name size $ ightarrow$ )
		ob= [] or [[]]
		$size = \{n\} or \{n m\}$
3B401	xRDZ	$(x \rightarrow )$
3ED22	xRECN	( name $ ightarrow$ )
		( $name \rightarrow$ )
0110AB		$(\  ightarrow\ )$
3ED56	xRECV	$(\  ightarrow\ )$
048314	~xREF	$( [[]] \rightarrow [[]]' )$
3B819	xRE	$((x,y) \rightarrow x)$
		$( [] \rightarrow []' )$
	~xREMAINDER	( p1 p2 $\rightarrow$ p3 )
	xRENAME	( name name' $\rightarrow$ )
	~xREORDER	( pol var → pol' )
	xREPEAT	$(1/0 \rightarrow)$
3B9D2	xREPL	( ob pos new → ob' )
		ob= [[]] or [] or {} or \$ or PICT
		$pos= N or \{n m\} or (n,m)$
3C41A		$(n_{int} \rightarrow )$
	xRESTORE	$(:port:name \rightarrow)$
	XRESULTANT	$(p1 p2 \rightarrow res)$
	xREVLIST	$(\{1n\} \rightarrow \{n1\}')$
3F070	xR>I	$(x \rightarrow n)$
005214	~ DIGG!!	UserRPL: R→I
00D314		$(f var \rightarrow F)$
0220AB	xRKFERR	$(\{\} h \rightarrow \{\} h \text{ dy err})$
	XRKFSTEP	$( \{\} \text{ tol } h \rightarrow \{\} \text{ tol } h' )$
0200AB	XKKF	( $\{\}\  xtol\  xTf \rightarrow \{\}\  xtol\ )$ ( $\{\}\  \{xtol\  step\}\  xTf \rightarrow \{\}\  xtol\ )$
20501	**DT	$(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $
38E01 38E21	xRL	$( \# \rightarrow \# ' )$
3AEB1	xRLB xRND	(
3B16C	xRNRM	$([] \rightarrow x)$
3DD33		$(n \dots 1 n \rightarrow 1 n \dots 2)$
3DD33		$(1n n \rightarrow 2n 1)$
06F0AB	xROMUPLOAD	$( \rightarrow )$
3D3CE	xROOT	$(\neg )$ ( prog/s var guess $\rightarrow x$ )
JUJCE	YIOOI	( prog/s var guess $\rightarrow x$ ) ( prog/s var {guesses} $\rightarrow x$ )
		( Prog/p var [Agepaca] , v )

Addr.	Name	Description
3DC71	xROT	$(123 \rightarrow 231)$
03C0AB	xROW-	( [[]] $nrow \rightarrow [[]]'$ [] )
		( [] n $ ightarrow$ []' elt )
03D0AB	xROW+	( [[]] [[]]' $n \rightarrow$ [[]]'' )
		( [[]] [] $n \rightarrow [[]]'$ )
		$( [] n n' \rightarrow [] )$
0360AB	$x \rightarrow ROW$	$([[]] \rightarrow [1][n] n)$
		$([] \rightarrow x1xn n)$
0370AB	$xROW \rightarrow$	$([1][n] n \rightarrow [])$
		$(x1xn \rightarrow [])$
38E41	xRR	$( \# \rightarrow x' )$
38E61	xRRB	$( \# \rightarrow x' )$
0340AB		$( [[]] \rightarrow [[]]' )$
	~xrref	( [[]] $\rightarrow$ [pp] [[]]')
078314	~xRREFMOD	$( [[]] \rightarrow [[]]' )$
	xRRKSTEP	( $\{\}$ xtol h last $\rightarrow$ $\{\}$ xtol h' cur )
0230AB		$\{ \} \text{ xtol xTfinal } \rightarrow \{ \} \text{ xtol } \}$
	xRSBERR	( $\{\}\ h \rightarrow \{\}\ h\ dy\ err\ )$
3B22F	xRSD	( [B] [[A]] [Z] $\rightarrow$ []')
		$( [[B]] [[A]] [[Z]] \rightarrow [[]]' )$
0400AB		( []/[[]] i j → []/[[]] )
3C9E5		$( ob1 ob2 \rightarrow 1/0 )$
3EE82		$(\rightarrow)$
3C444	x*H	$(xf \rightarrow )$
		UserRPL: SCALEH
3C464	x*W	$(yf \rightarrow )$
		UserRPL: SCALEW
3C4D5		$(xs ys \rightarrow )$
3E1EF	xSCATRPLOT	$(\rightarrow)$
0330AB	xSCHUR	$( \hspace{.1cm} [\hspace{.1cm} [\hspace{.1cm} [\hspace{.1cm}]\hspace{.1cm}] \hspace{.1cm} \rightarrow \hspace{.1cm} [\hspace{.1cm} [\hspace{.1cm} Q\hspace{.1cm}] \hspace{.1cm}] \hspace{.1cm} [\hspace{.1cm} [\hspace{.1cm} T\hspace{.1cm}] \hspace{.1cm}] \hspace{.1cm})$
3B5BA	xSCI	$(n \rightarrow )$
3E127	xSCLSIGMA	$(\hspace{1em}  o \hspace{1em})$ UserRPL: SCL $\Sigma$
20205	CCONT	( name $\rightarrow$ )
3E385	xSCONJ ~xSCROLL	$( name \rightarrow )$ $( ob \rightarrow )$
	xSCROLL xSDEV	$(op \rightarrow)$ $(\rightarrow xsdev)$
3DF32	YOUFA	$( \rightarrow xsaev )$ $( \rightarrow [x1xn] )$
		$( \rightarrow [x_1, \dots x_{II}] )$

```
Addr.
           Name
                                Description
3ECB0
           xSEND
                                (\text{name} \rightarrow )
                                ( \{names\} \rightarrow )
                                ( \{\{\text{old new}\}...\} \rightarrow \}
                                ( prog var start end incr \rightarrow {} )
0530AB
           xSEQ
007314
           ~xSERIES
                                (func var order \rightarrow {} symb')
3ED91
           xSERVER
                                ( \rightarrow )
                                ( symb \rightarrow symb' )
064314
           ~xSEVAL
3B4C9
                                (n \rightarrow )
           xSF
3E696
                                ( symb name \rightarrow symb' )
           xSHOW
                                ( symb {names} \rightarrow symb')
0630AB
           xSIDENS
                                (x \rightarrow x')
                                ( f var \rightarrow F )
0020DE
           xSIGMA
0010DE
           xSIGMAVX
                                (f(x) \rightarrow F(x))
05F314
           ~xSIGNTAB
                                (symb \rightarrow \{\})
                                (x \rightarrow x')
3A3EE
           xSIGN
033314
           ~xSIMP2
                                ( x y \rightarrow x/gcd y/gcd)
                                (symb \rightarrow symb')
0220DE
          xSIMPLIFY
018314
           ~xSINCOS
                                (symb \rightarrow symb')
3A678
           xSINH
                                (x \rightarrow x')
3E331
                                ( name \rightarrow )
           xSINV
                                (x \rightarrow x')
3A57C
           xSIN
3BB1F
           xSIZE
                                ( ob \rightarrow n )
                                ( ob \rightarrow {N m})
38E81
           xSL
                                ( # → #')
                                ( # → #')
38EA1
           xSLB
                                ( \rightarrow )
00C0AB
           xSLOPEFIELD
                                ( name \rightarrow )
3E35B
           xSNEG
                                ( [] \rightarrow x)
0290AB
          xSNRM
03F314
           ~xSOLVE
                                ( symb var \rightarrow {zeros} )
                                ( \rightarrow )
086314
           ~xSOLVER
                                ( symb \rightarrow {zeros} )
008314
           ~xSOLVEVX
                                ( \{\} \rightarrow \{\}')
05E0AB
          xSORT
0130AB
           xSPHERE
                                ( \rightarrow )
                                (x \rightarrow x')
3A4EF
           xSQ
                                ( # → #')
38EC1
           xSR
0280AB
                                ( [[]] \rightarrow x )
          xSRAD
38EE1
           xSRB
                                ( # → #')
3EC55
           xSRECV
                                (n \rightarrow \$ 0/1)
                                ( str find repl \rightarrow str' )
0100DD
          xSREPL
```

Addr.	Name	Description
381AB	xSTART	( start finish $ ightarrow$ )
3B5FA	xSTD	$( \rightarrow )$
3851F	xSTEP	$(n \rightarrow )$
		( symb $ ightarrow$ )
3D3AE	xSTEQ	$( ob \rightarrow )$
3EE62	xSTIME	$(x \rightarrow )$
39164	xSTOALARM	( time $\rightarrow$ n )
		( $\{  ext{date time act rep} \}  ightarrow  ext{n}$ )
3B749	xSTOF	( { $\#$ s1 $\#$ u1 $\#$ s2 $\#$ u2} $ ightarrow$ )
3DD6E	xSTOSIGMA	$( ob \rightarrow )$
		UserRPL: STO $\Sigma$
0400DE	xSTOVX	( name $ ightarrow$ )
		Store object into the reserved CAS variable
		VX.
3EF07	xSTOKEYS	$(\{ob\ key\ \ldots\}\  o\ )$
		( {'S' ob key $\dots$ } $\rightarrow$ )
		$( 'S' \rightarrow )$
3E739	xSTO	( ob name $\rightarrow$ )
		( ob :port:name $\rightarrow$ )
		( lib port $\rightarrow$ )
		( ob 'name(i)' $\rightarrow$ )
3E823	xSTO>	$( ob id \rightarrow )$
		$(ob symb \rightarrow)$
		Like xSTO, but if the level 1 argument is sym-
		bolic, use the first element of it as the variable
27406	G.T.O.	to write to.
3E406	xSTO-	$( ob name \rightarrow )$
3E46C	xSTO/	$( ob name \rightarrow )$
3E4D2	xSTO*	( ob name $\rightarrow$ ) ( ob name $\rightarrow$ )
3E3AF		( $\$ \rightarrow ob$ )
3BBD9	xSTR>	$(S \rightarrow OD)$ UserRPL: STR $\rightarrow$
0580AB	xSTREAM	( $\{\} \text{ prog } \rightarrow \mathbf{x} \)$
3BBBE	x>STREAM x>STR	$(\ \ \ ) \ \ )$
SDDDE	X/SIK	$(OD \rightarrow S)$ $UserRPL: \rightarrow STR$
3B6C1	xSTWS	( $n \rightarrow$ )
		· · · · · · · · · · · · · · · · · · ·
7007	25000	
3B8D7	xSUB	( ob start end → ob' ) ob= [[]], \$, {}, grob start,end = n, {n m}, (n,m)

```
Addr.
           Name
                                Description
002314
                                ( symb var=s1 \rightarrow symb' )
           ~xSUBST
                                ( x1 x2 \rightarrow x3 )
06F314
           ~xSUBTMOD
02E0AB
                                ( [[]] \rightarrow [[U]] [[V]] [S] )
          xSVD
                                ( [[]] \rightarrow [] )
02F0AB
           xSVL
3DC20
                                ( ob1 ob2 \rightarrow ob2 ob1 )
           xSWAP
04E314 ~xSYLVESTER
                                ( [[]] \rightarrow [D] [P] )
39705
           xSYSEVAL
                                ( \# \rightarrow ? )
061314
          ^{\sim}xTABVAL
                                (symb(x) \{vals\} \rightarrow symb(x) \{\{vals\}\}
                                {res}} )
                                ( symb(x) \rightarrow symb(x) \{\{\}\}\} grob )
060314
           ~xTABVAR
                                ( ob tag \rightarrow :tag:ob )
3EFB1
           x->TAG
                                UserRPL: \rightarrow TAG
0520AB
          xTAIL
                                ( \{\} \rightarrow \{\}')
                                ( \$ \rightarrow \$' )
01C0DE
                                (symb \rightarrow symb')
         xTAN2CS2
021314
           ~xTAN2SC2
                                (symb \rightarrow symb')
                                (symb \rightarrow symb')
01F314 ~xTAN2SC
                                (x \rightarrow x')
3A70C
           xTANH
3A624
           xTAN
                                (x \rightarrow x')
                                ( symb \rightarrow symb' )
006314
          ~xTAYLOR0
                                ( symb var n \rightarrow symb' )
3E6CA
          xTAYLR
05B314
          ~xTCHEBYCHEFF
                                (n \rightarrow pol)
01A314
                                ( symb \rightarrow symb' )
           ~xTCOLLECT
0640AB xTDELTA
                                (xy \rightarrow x')
                                ( ob \rightarrow ? time )
065314
           ~xTEVAL
                                (symb \rightarrow symb')
013314 ~xTEXPAND
                                ( \rightarrow )
3C8FA
           xTEXT
37F7F
                                (0/1 \rightarrow )
           XTHEN
                               ( \rightarrow # )
39093
           xTICKS
                                ( time \rightarrow )
39124
           xSETTIME
                                UserRPL: \to \texttt{TIME}
                                (\rightarrow \text{time})
3905D
           xTIME
                                (xy \rightarrow x')
0650AB
           xTINC
                                ((x1,y1)(x2,y2) \rightarrow )
3C6B6
           XTLINE
                                ( \{\#n1 \#m1\} \{\#n2 \#m2\} \rightarrow )
                                (symb \rightarrow symb')
019314
           ~xTLIN
3E97B
           xTMENU
                                ( % \rightarrow [InitMenu%] )
                                ( Ob \rightarrow [@LIST InitMenu] )
```

Addr.	Name	Description
3DF4D	xTOT	( $ ightarrow$ xsum )
		$( \rightarrow \{x1xn\} )$
0270AB	XTRACE	$( [[]] \rightarrow x )$
045314	~xTRAN	$( \hspace{.1cm} [\hspace{.1cm} [\hspace{.1cm}]\hspace{.1cm}] \hspace{.1cm} \rightarrow \hspace{.1cm} [\hspace{.1cm} [\hspace{.1cm}]\hspace{.1cm}] \hspace{.1cm}' \hspace{.1cm} )$
		( name $ ightarrow$ )
3EE0C	xTRANSIO	$(n \rightarrow )$
01C314	~xTRIGCOS	( $symb \rightarrow symb'$ )
082314	~xTRIGO	$( \rightarrow )$
01D314	$\sim$ xTRIGSIN	$(symb \rightarrow symb')$
01B314	~xTRIG	$(symb \rightarrow symb')$
01E314	$\sim$ xTRIGTAN	$(symb \rightarrow symb')$
3C084	xTRN	$( [[]] \rightarrow [[]]' )$
		( name $\rightarrow$ )
3AF3E	xTRNC	$(x n \rightarrow )$
063314	~xTRUNC	( $symb1 symb2 \rightarrow symb3$ )
3C99D	xTRUTH	$( \rightarrow )$
015314	~xTSIMP	$(symb \rightarrow symb')$
391F8	xTSTR	( date time $ ightarrow$ \$ )
39456	xTVARS	( $ntype \rightarrow \{\}$ )
		$( \{n\} \rightarrow \{\} )$
0470AB	XTVM	$( \rightarrow )$
0480AB	xTVMBEG	$( \rightarrow )$
0490AB	XTVMEND	$( \rightarrow )$
	xTVMROOT	$( var \rightarrow x )$
3B2DC	x%T	$(xy \rightarrow 100y/x)$
3BC39	xTYPE	( ob $ ightarrow$ %type )
	XUBASE	$(u \rightarrow u')$
	XUFACT	$(u1 u2 \rightarrow u3)$
0140DD	$\mathtt{xUFL1} { ightarrow} \mathtt{MINIF}$	$( ob n \rightarrow font )$
38FB5	x>UNIT	$(x u \rightarrow u')$
		UserRPL:  o UNIT
3F249	xUNPICK	$(obnob1 ob n \rightarrow obob2)$
3F22E	xUNROT	$(123 \rightarrow 312)$
38195	XUNTIL	$(\rightarrow)$
39420	xUPDIR	$(\  ightarrow\ )$
3E07D	xUTPC	$(n \times \rightarrow x')$
3E0BD	xUTPF	( n1 n2 x $\rightarrow$ x' )
3E09D	xUTPN	$(n \vee x \rightarrow x')$
3E0DD	XUTPT	$(n \times \rightarrow x')$

Addr.	Name	Description
38F81	xUVAL	$(u \rightarrow x)$
3C2AC	$\times V >$	$( []/() \rightarrow x y )$
		$( []/() \rightarrow x y z )$
		(in current co-system)
		$\textbf{UserRPL:} \ \mathtt{V} {\rightarrow}$
3C2D6	x>V2	$(xy \rightarrow [])$
		$(xy \rightarrow ())$
		UserRPL: $\rightarrow$ V2
3C30A	x>V3	$(xyz \rightarrow [])$
		UserRPL: $\rightarrow$ V3
053314	$\sim$ xVANDERMONDE	$( \{\} \rightarrow [[]] )$
3943B	xVARS	$( \rightarrow \{\} )$
3DF68	xVAR	$( \rightarrow x )$
		$( \rightarrow [x1xn] )$
08C314	~xVER	$( \rightarrow \ \ \ \ \ )$
	xVERSION	$( \rightarrow \$ \$)$
00A0DD	xVISITB	( name $\rightarrow$ )
	xVISIT	( name $\rightarrow$ )
3DB04	xMATCHDN	( symb {spat srepl} $ ightarrow$ symb' 0/1 )
		( symb {spat srepl scond} $\rightarrow$ symb'
		0/1)
		UserRPL: \MATCH
	xVTYPE	$(name \rightarrow n)$
3A442	xSQRT	$(x \rightarrow x')$
		UserRPL: $\sqrt{}$
39819	XWAIT	$($ sec $\rightarrow$ $)$
		$(0 \rightarrow \text{rc.p})$
	xWHILE	$( \rightarrow )$
	xWIREFRAME	$( \rightarrow )$
	xWSLOG	$( \rightarrow \$ \$ \$ \$ )$
3ABAF	xFACT	$(x \rightarrow x')$
2=025	*****	UserRPL: !
3E03D	xXCOL	$(n \rightarrow )$
0700AB	XXGET	$(\text{name} \rightarrow)$
3EC35	XXMIT	$($\Rightarrow 1)$
067214	~>/	$(\$ \rightarrow \$ \text{rest 0})$
067314	~xXNUM	$(x \rightarrow x')$

Addr.	Name	Description
3CB7A	xXOR	( # #' → #'' )
		( \$ \$' \rightarrow \$'' )
		$(1/01/0 \rightarrow 1/0)$
3AD65	xXPON	$( \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ )$
		( symb $ ightarrow$ )
0710AB	xXPUT	( name $ ightarrow$ )
068314	$\sim$ xXQ	$(x \rightarrow x')$
0500AB	xXRECV	( name $ ightarrow$ )
3C915	xXRNG	$(x1 x2 \rightarrow )$
3A278	xXROOT	$(y x \rightarrow Y')$
06E0AB	xXSERV	$( \rightarrow )$
04F0AB	xXSEND	( name $ ightarrow$ )
0000AB	xXVOL	( x1 x2 $\rightarrow$ )
0030AB	xXXRNG	( x1 x2 $\rightarrow$ )
39CFC	X-	$(xy \rightarrow x-y)$
39F49	x/	$(xy \rightarrow x/y)$
39DE8	x*	$(xy \rightarrow x*y)$
3CF80	x<=?	$(xy \rightarrow 1)$
		$(xy \rightarrow 0)$
		$\text{UserRPL:} \leq$
3CD21	x#?	$(xy \rightarrow 1)$
		$(xy \rightarrow 0)$
		$\text{UserRPL:} \neq$
3D01F	x>=?	$(xy \rightarrow 1)$
		$(xy \rightarrow 0)$
		$\text{UserRPL:} \geq$
39B58	X+	$(xy \rightarrow x+y)$
3CE42	x<	$(xy \rightarrow 1)$
		$(xy \rightarrow 0)$
3CBF6	x==	$(xy \rightarrow 1)$
		$(xy \rightarrow 0)$
3CEE1	x>	$(xy \rightarrow 1)$
		$(xy \rightarrow 0)$
398B9	x=	$(xy \rightarrow x=y)$
3B251	x%	$(xy \rightarrow xy/100)$
3E05D	xYCOL	$(n \rightarrow )$
3C935	xYRNG	$(y1 y2 \rightarrow )$
00B0AB	xYSLICE	$(\rightarrow)$
0010AB	xYVOL	( y1 y2 $\rightarrow$ )

Addr.	Name	Description
3A097	x^	( y x → y^x )
0040AB	xYYRNG	( y1 y2 $\rightarrow$ )
040314	~xZEROS	( symb var $ ightarrow$ {zeros} )
05F0AB	xZFACTOR	( $xTr yPr \rightarrow xZf$ )
0020AB	xZVOL	( x1 x2 $\rightarrow$ )

# **Appendix D Library 256 and EXTABLE**

### **D.1** The Development Library 256

Library 256 is built-in in the calculator, and contains several commands for the programmers. Some of this commands are described in section A.3. The list belows contains the commands and their rompointer address, should you want to use one of them in your program.

Addr.	Name	Description
000100	x→H	( ob $ ightarrow$ \$hex )
001100	$xH{\longrightarrow}$	( $hex \rightarrow ob$ )
002100	$x{ ightarrow} A$	( ob $\rightarrow$ hxs )
003100	$x A \!\!\to\!\!$	$( \text{hxs} \rightarrow \text{ob} )$
004100	$xA \rightarrow H$	$( \text{hxs} \rightarrow \text{hex} )$
005100	$xH \rightarrow A$	( $hex \rightarrow hxs$ )
006100	$x{ ightarrow}CD$	( $hex \rightarrow code$ )
007100	$\texttt{xCD}{\rightarrow}$	( code $ ightarrow$ \$hex )
008100	$xS{ ightarrow} H$	( $\$ \rightarrow \$$ hex )
009100	$xH \rightarrow S$	( $\$$ hex $ o$ $\$$ )
00A100	$x{ ightarrow} LST$	( comp $\rightarrow$ {} )
		( oblobn %n $\rightarrow$ {} )
00B100	$x{ ightarrow}ALG$	$($ comp $\rightarrow$ symb $)$
		(oblobn %n $ ightarrow$ symb)
00C100	$x{ ightarrow} PRG$	$(comp \rightarrow ::)$
		(oblobn %n $\rightarrow$ ::)
00D100	$\texttt{xCOMP}{\rightarrow}$	$(comp \rightarrow oblobn %n)$
00E100	$x \rightarrow RAM$	$( ob \rightarrow ob )$
00F100	xSREV	(\$ \rightarrow\$')
010100	xPOKE	( hxs $hex \rightarrow$ )
011100	xPEEK	( hxs1 hxs2 $\rightarrow$ \$hex )
012100	xAPEEK	$( hxs \rightarrow hxs' )$

Addr.	Name	Description
013100	xR~SB	( % → # )
		( ♯ → % )
014100	xSB~B	$( \# \rightarrow hxs )$
		( hxs $\rightarrow$ # )
015100	xLR~R	( %% → % )
		$( % \rightarrow % )$
016100	xS~N	( $\$ \rightarrow \mathtt{ID}$ )
		( ID $\rightarrow$ \$ )
017100	xLC~C	( %%C → %C )
		( %C → %%C )
018100	$\texttt{xASM}{\rightarrow}$	( Code $\rightarrow$ \$ )
019100	xBetaTesting	$( \rightarrow \$ )$
01A100	xCRLIB	$(\rightarrow lib)$
01B100	xCRC	$( \$ \rightarrow \# crc )$
01C100	xMAKESTR	( xlen $\rightarrow$ \$ )
01D100	xSERIAL	$( \rightarrow \$ )$
01E100	xASM	$( \ \$ \rightarrow ob \ )$
01F100	xER	( $\$$ {errors} $\rightarrow$ $\$$ ' )
020100	x→S2	( ob $\rightarrow$ \$ )
021100	xXLIB~	( xlib xn $\rightarrow$ ROMPTR )
		( ROMPTR $ ightarrow$ xlib xn )

## **D.2** The EXTABLE Library

The EXTABLE library contains the table of entry points and their addresses. It contains some commands which might be useful in programs, also listed below. For a description of these commands, see section A.1.

Addr.	Name	Description
001102	xGETADR	$( \ \$ \rightarrow \text{hxs} \ )$
		Get the address of an entry name.
002102	xGETNAME	( hxs $\rightarrow$ \$ )
		Get the entry name corresponding to an address.
003102	xGETNAMES	( $\$$ start $\rightarrow$ $\{\}$ )
		Get all entry names which start with the given
		string.
004102	xGETNEAR	$(\$sub \rightarrow \{\})$
		Get all entry names which contain the given string.

# **Appendix E Error Messages**

This appendix lists all "error" messages in the HP49G, even if most have nothing to do with errors. It is possible to generate an error with the following messages directly, with words such as ERRORSTO or ERROROUT, or recall them to the stack with JstGetTHEMESG. See Chapter 22.

All numbers listed are in hexadecimal format.

The symbol → represents a line break.

#n	Message	#n	Message
1	Insufficient Memory	1D	Erase Fail, Locked Block
2	Directory Recursion	1E	Write Adr outside ROM
3	Undefined Local Name	1F	Write Fail, Rom Faulty
4	Undefined XLIB Name	20	Write Fail, Low bats
5	Memory Clear	21	Write Fail, Locked Block
6	Power Lost	101	No Room to Save Stack
7	Warning:	102	Can't Edit Null Char.
8	Invalid Card Data	103	Invalid User Function
9	Object In Use	104	No Current Equation
Α	Port Not Available	106	Invalid Syntax
В	No Room in Port	107	Real Number
С	Object Not in Port	108	Complex Number
D	Recovering Memory	109	String
E	Try To Recover Memory?	10A	Real Array
F	Replace RAM, Press ON	10B	Complex Array
10	No Mem To Config All	10C	List
11	Undefined FPTR Name	10D	Global Name
12	Invalid Bank Data	10E	Local Name
13	Full Check Bad Crc	10F	Program
14	Cmprs: not a user bank	110	Algebraic
15	No or 2 system bank	111	Binary Integer
16	Invalid bank	112	Graphic
17	Invalid bank number	113	Tagged
18	Inexisting pack	114	Unit
19	Pack twice	115	XLIB Name
1A	Ins. Mem.	116	Directory
1B	Erase Fail, Rom faulty	117	Library
1C	Erase Fail, Low bats	118	Backup

#n	Message	#n	Message
119	Function	14A	Extended 2
11A	Command	14B	Extended 3
11B	System Binary	14C	YES
11C	Long Real	14D	NO
11D	Long Complex	14E	TRUE
11E	Linked Array	14F	FALSE
11F	Character	150	Are you sure?
120	Code	151	Low Memory Condition →Please Wait
121	Library Data	152	CATALOG
122	External	153	Nonexistent Find Pattern
124	LAST STACK Disabled	154	Not Found
125	LAST CMD Disabled	155	Nonexistent Replace Pattern
126	HALT Not Allowed	156	Can't Find Selection
127	Array	157	Y= not available
128	Wrong Argument Count	158	Warning:→Changes will not be saved
129	Circular Reference	159	Result not editable in EQW
12A	Directory Not Allowed	201	Too Few Arguments
12B	Non-Empty Directory	202	Bad Argument Type
12C	Invalid Definition	203	Bad Argument Value
12D	Missing Library	204	Undefined Name
12E	Invalid PPAR	205	LASTARG Disabled
12F	Non-Real Result	206	Incomplete⇒Subexpression
130	Unable to Isolate	207	Implicit () off
131	No Room to Show Stack	208	Implicit () on
132	Warning:→	301	Positive Underflow
133	Error:	302	Negative Underflow
134	Purge?	303	Overflow
135	Out of Memory	304	Undefined Result
136	Stack	305	Infinite Result
137	Last Stack	501	Invalid Dimension
138	Last Commands	502	Invalid Array Element
139	Key Assignments	503	Deleting Row
13A	Alarms	504	Deleting Column
13B	Last Arguments	505	Inserting Row
13C	Name Conflict	506	Inserting Column
13D	Command Line	601	Invalid $\Sigma$ Data
13F	Interrupted	602	Nonexistent $\Sigma DAT$
140	Integer	603	Insufficient $\Sigma$ Data
	0	604	Invalid ΣPAR
141	Symbolic Matrix		Invalid $\Sigma$ Pata LN(Neg)
142	Font	605	Invalid $\Sigma$ Data LN(Neg) Invalid $\Sigma$ Data LN(0)
143	Aplet	606	
144	Extended Real	607	Invalid EQ
145	Extended Complex FlashPtr	608	Current equation:
146		609	No current equation.
147	Extended Ptr MiniFont	60A	Enter eqn, press NEW
148		60B	Name the equation, ⇒press ENTER
149	Extended 1	60C	Select plot type

#n	Message	#n	Message
60D	Empty catalog	70D	Grads
60E	undefined	70E	Rectangular
60F	No stat data to plot	70F	Polar
610	Autoscaling	710	Spherical
611	Solving for	711	Operating Mode
612	No current data. Enter	712	Number Format
613	data point, press $\Sigma$ +	713	Angle Measure
614	Select a model	714	Coord System
615	No alarms pending.	715	FM,
616	Press ALRM to create	716	Beep
617	Next alarm:	717	Key Click
618	Past due alarm:	718	Last Stack
619	Acknowledged	719	Choose calculator operating mode
61A	Enter alarm, press SET	71A	Choose number display format
61B	Select repeat interval	71B	Choose decimal places to display
61C	I/O setup menu	71C	Choose angle measure
61D	Plot type:	71D	Choose coordinate system
61E	""	71E	Use comma as fraction mark?
61F	(OFF SCREEN)	71F	Enable standard beep?
620	Invalid PTYPE	720	Enable key click?
621	Name the stat data, → press ENTER	721	Save last stk for UNDO and ANS?
622	·	721	CALCULATOR MODES
623	Enter value (zoom out➡if >1), press ENTER Copied to stack	723	Font:
	x axis zoom w/AUTO.⇒	_	Stack:
624 625	x axis zoom. ₩/AU 1 O. ➡	724 725	Small
626		726	Textbook
627	y axis zoom.⇒ x and y axis zoom.⇒	727	Edit:
628	X and y axis zoom.→	728	Small
	IR/wire:		
629	ASCII/binary:	729	Full Page
62A	baud:	72A	Indent
62B	parity:	72B	EQW:
62C	checksum type:	72C	Small
62D	translate code:	72D	Small Stack Disp
62E	Enter matrix, then NEW	72E	Header:
62F	No Associated Numeric View	72F	Clock
701	Algebraic	730	Analog
702	RPN	731	Choose system font
703	Standard	732	Display stack using small font?
704	Std	733	Use pretty print in the stack?
705	Fixed	734	Edit using small font?
706	Fix	735	Edit in full page?
707	Scientific	736	Automatically indent new lines?
708	Sci	737	Edit in EQW using small font?
709	Engineering	738	Display EQW using small font?
70A	Eng	739	Choose header height
70B	Degrees	73A	Display ticking clock?
70C	Radians	73B	Analog clock?

#n	Message	#n	Message
73C	DISPLAY MODES	76B	Conic
73D	Indep var:	76C	Truth
73E	Modulo:	76D	Histogram
73F	Verbose	76E	Bar
740	Step/Step	76F	Scatter
741	Complex	770	Slopefield
742	Approx	771	Fast3D
743	Incr Pow	772	Wireframe
744	Simp Non-Rational	773	Ps-Contour
745	Rigorous	774	Y-Slice
746	Numeric	775	Gridmap
747	Enter independent variable name	776	Pr-Surface
748	Enter modulo value	777	Deg
749	Display calculus information?	778	Rad
74A	Perform operations step by step?	779	Grad
74B	Allow complex numbers?	77A	Type:
74C	Perform approx calculations?	77B	∡:
74D	Increasing polynomial ordering?	77C	EQ:
74E	Simplify non rational expr?	77D	Indep:
74F	Don't simplify  X  to X?	77E	Connect
750	Replace constants by values?	77F	Simult
751	CAS MODES	780	H-Tick:
752	Goto row:	781	V-Tick:
753	Goto column:	782	Pixels
754 756	Specify a row to go to	783	Depnd: Save Animation
755 756	Specify a column to go to Matrix Writer	784	ΣDAT:
756 757		785	Col:
757	Bad range value	786	
758	Start:	787	Cols: F:
759	Step:	788	
75A	Type:	789	H-Var:
75B	Zoom:	78A	V-Var:
75C	Small Font	78B	Stiff
75D	File:	78C	∂F∂Y:
75E	Enter starting value	78D	$\partial F \partial T$ :
75F	Enter increment value	78E	Choose type of plot
760	Choose table format	78F	Choose angle measure
761	Enter zoom factor	790	Enter function(s) to plot
762	Display table using small font?	791	Enter independent variable name
763	Enter a filename to save data	792	Connect plot points?
764	TABLE SETUP	793	Plot functions simultaneously?
765	Automatic	794	Enter horizontal tick spacing
766	Build Your Own	795	Enter vertical tick spacing
767	Function	796	Tick spacing units are pixels?
768	Polar	797	Enter dependent variable name
769	Parametric	798	Save slices animation?
76A	Diff Eq	799	Enter data to plot

#n	Message	#n	Message
79A	Enter col to use for horizontal	7C9	Enter minimum Z view-volume val
79в	Enter col to use for vertical	7CA	Enter maximum Z view-volume val
79C	Enter horizontal variable	7CB	Enter X eyepoint coordinate
79D	Enter vertical variable	7CC	Enter Y eyepoint coordinate
79E	Use stiff diff eq solver?	7CD	Enter Z eyepoint coordinate
79F	Enter derivative w.r.t. soln	7CE	Enter absolute error tolerance
7A0	Enter derivative w.r.t. indep	7CF	Enter minimum XX range value
7A1	PLOT SETUP	7D0	Enter maximum XX range value
7A2	H-View:	7D1	Enter minimum YY range value
7A3	V-View:	7D2	Enter maximum YY range value
7A4	Indep Low:	7D3	PLOT WINDOW
7A5	High:	7D4	Default
7A6	Step:	7D5	FUNCTION
7A7	Pixels	7D6	POLAR
7A8	Depnd Low:	7D7	PARAMETRIC
7A9	High:	7D8	DIFF EQ
7AA	X-Left:	7D9	CONIC
7AB	X-Right:	7DA	TRUTH
7AC	Y-Near:	7DB	HISTOGRAM
7AD	Y-Far:	7DC	BAR
7AE	Step Indep:	7DD	SCATTER
7AF	Depnd:	7DE	SLOPEFIELD
7в0	Bar Width:	7DF	FAST3D
7B1	Z-Low:	7E0	WIREFRAME
7B2	Z-High:	7E1	PS-CONTOUR
7B3	XE:	7E2	Y-SLICE
7B4	YE:	7E3	GRIDMAP
7B5	ZE:	7E4	PR-SURFACE
7в6	Init:	7E5	PLOT WINDOW
7в7	Final:	7E6	Enter minimum X view-volume val
7в8	Init-Soln:	7E7	Enter maximum X view-volume val
7B9	Tol:	7E8	Enter minimum Y view-volume val
7BA	XXLeft:	7E9	Enter maximum Y view-volume val
7BB	XXRight:	7EA	Enter indep var sample count
7BC	YYNear:	7EB	Enter depnd var sample count
7BD	YYFar:	7EC	Goto Level:
7BE	Enter minimum horizontal value	7ED	Specify a level to go to
7BF	Enter maximum horizontal value	7EE	HISTORY
7C0	Enter minimum vertical value	801	Must be $\geq 0$
7C1	Enter maximum vertical value	802	Must be bewteen 0 and 1
7C2	Enter minimum indep var value	803	$\mu$ 0:
7C3	Enter maximum indep var value	804	$\bar{x}$ :
7C4	Enter indep var increment	805	N:
7C5	Indep step units are pixels?	806	$\alpha$ :
7C6	Enter minimum depend var value	807	$\sigma$ :
7C7	Enter maximum depend var value	808	Null hypothesis population mean
7C8	Enter bar width	809	Sample mean

#n	Message	#n	Message
80A	Sample Size	839	Sample Mean
80B	Significance level	83A	Significance level
80C	Population standard deviation	83B	Sample size
80D	Z-TEST: 1 $\mu$ , KNOWN $\sigma$	83C	T-TEST: 1 $\mu$ , UNKNOWN $\sigma$
80E	Alternative Hypothesis	83D	$ar{x}$ 1:
80F	$ar{x}1$ :	83E	S1:
810	$\sigma$ 1:	83F	N1:
811	N1:	840	$\alpha$ :
812	$\alpha$ :	841	$ar{x}2$ :
813	$\bar{x}2:$	842	S2:
814	$\sigma 2$ :	843	N2:
815	N2:	844	Pooled?
816	Sample mean for population 1	845	Sample mean for population 1
817	Std deviation for population 1	846	Std deviation for sample 1
818	Sample size for population 1	847	Sample size for population 1
819	Significance level	848	Significance level
81A	Sample mean for population 2	849	Sample mean for population2
81B	Std deviation for population 2	84A	Std deviation for sample 2
81C	Sample size for population 2	84B	Sample size for population 2
81D	Z-TEST: 2 $\mu$ , KNOWN $\sigma$	84C	"Pooled" if checked
81E	$\pi 0$ :	84D	T-TEST: 2 $\mu$ , UNKNOWN $\sigma$
81F	x:	84E	$\bar{x}$ :
820	N:	84F	$\sigma$ :
821	$\alpha$ :	850	N:
822	Null hyp. population proportion	851	C:
823	Success count	852	Sample mean
824	Sample size	853	Population standard deviation
825	Significance level	854	Sample size
826	Z-TEST: 1 P	855	Confidence level
827	X1:	856	CONF. INT.: 1 $\mu$ , KNOWN $\sigma$
828	N1:	857	$\bar{x}1$ :
829	$\alpha$ :	858	$\sigma 1$ :
	χ2:		N1:
82A	N2:	859	C:
82B		85A	$ar{x}2$ :
82C	Success count for sample 1	85B	
82D	Size of sample 1	85C	$\sigma 2$ :
82E	Significance level	85D	N2:
82F	Success count for sample 2	85E	Sample mean for population 1
830	Size of sample 2	85F	Std deviation for sample 1
831	Z-TEST: 2 P	860	Size of sample 1
832	$ar{x}$ :	861	Sample mean for population 2
833	Sx:	862	Std deviation for sample 2
834	$\mu$ 0:	863	Size of sample 2
835	α:	864	Confidence level
836	N:	865	CONF. INT.: 2 $\mu$ , KNOWN $\sigma$
837	Null hypothesis population mean	866	X:
838	Sample Standard deviation	867	N:

#n	Message	#n	Message
868	C:	897	Enter replace pattern
869	Sample success count	898	Case sensitive search?
86A	Sample size	899	Enter search pattern
86B	Confidence level	89A	FIND REPLACE
86C	CONF. INT.: 1 P	89B	FIND
86D	$\bar{x}1$ :	89C	Goto Line:
86E	N1:	89D	Specify a line to go to
86F	C:	89E	GOTO LINE
870	$ar{x}2$ :	89F	Goto Position:
871	N2:	8A0	Specify a position to go to
872	Sample 1 success count	8A1	GOTO POSITION
873	Sample 1 size	8A2	H-Factor:
874	Sample 2 success count	8A3	V-Factor:
875	Sample 2 size	8A4	Recenter on cursor
876	Confidence level	8A5	Enter horizontal zoom factor
877	CONF. INT.: 2 P	8A6	Enter vertical zoom factor
878	$\bar{x}$ :	8A7	Recenter plot on cursor?
879	Sx:	8A8	ZOOM FACTOR
87A	N:	8A9	Object:
87B	C:	8AA	Name:
87C	Sample mean	8AB	Directory
87D	Sample standard deviation	8AC	Enter New Object
87E	Sample size	8AD	Enter variable name
87F	Confidence level	8AE	Create a new directory?
880	CONF. INT.: 1 $\mu$ , UNKNOWN $\sigma$	8AF	NEW VARIABLE
881	$\bar{x}1$ :	8B0	Select Object
882	S1:	901	[not shown because too long]
883	N1:	902	[not shown because too long]
884	C:	903	[not shown because too long]
885	$ar{x}2$ :	904	[not shown because too long]
886	S2:	905	[not shown because too long]
887	N2:	906	[not shown because too long]
888	Pooled	907	[not shown because too long]
889	Sample 1 mean	908	[not shown because too long]
88A	Std deviation for sample 1	909	[not shown because too long]
88B	Sample 1 size	90A	[not shown because too long]
88C	Sample 2 mean	90B	[not shown because too long]
88D	Std deviation for sample 2	90C	[not shown because too long]
			Inconclusive result
88E	Sample 2 size Confidence level	90D	Bad Guess(es)
88F		A01	Constant?
890	Pooled if checked	A02	
891	CONF. INT.: 2 $\mu$ , UNKNOWN $\sigma$	A03	Interrupted
892	Search for:	A04	Zero
893	Replace by: Case Sensitive	A05	Sign Reversal
894		A06	Extremum
895	Search For:	A07	Left
896	Enter search pattern	A08	Right

#n	Message	#n	Message
A09	Expr	В912	# Binary int
B01	Invalid Unit	В913	_Unit object
B02	Inconsistent Units	В914	Invalid object type
C01	Bad Packet Block Check	В915	Invalid object value
C02	Timeout	В916	Calculator Modes
C03	Receive Error	в917	Number Format:
C04	Receive Buffer Overrun	В918	Angle Measure:
C05	Parity Error	В919	Coord System:
C06	Transfer Failed	B91A	Beep
C07	Protocol Error	в91в	Clock
C08	Invalid Server Cmd.	B91C	FM,
C09	Port Closed	B91D	Choose number display format
COA	Connecting	B91E	Enter decimal places to display
C0B	Retry #	B91F	Choose angle measure
COC	Awaiting Server Cmd.	в920	Choose coordinate system
COD	Sending_	B921	Enable standard beep?
COE	Receiving	в922	Display ticking clock?
COF	Object Discarded	в923	Use comma as fraction mark?
C10	Packet #	B924	Standard
C11	Processing Command	B925	Std
C12	Invalid IOPAR	B926	Fixed
C13	Invalid PRTPAR	B927	Fix
C14	Low Battery	в928	Scientific
C15	Empty Stack	в929	Sci
C16	Row_	B92A	Engineering
C17	Invalid Name	B92B	Eng
D01	Invalid Date	B92C	Degrees
D02	Invalid Time	B92D	Deg
D03	Invalid Repeat	B92E	Radians
D04	Nonexistent Alarm	B92F	Rad
В901	Press [CONT] for menu	в930	Grads
В902	reset/delete this field	в931	Grad
в903	Reset value	в932	Rectangular
в904	Delete value	в933	Polar
в905	Reset all	в934	Spherical
B906	Valid object types:	в935	SYSTEM FLAGS
в907	Valid object type:	в936	01 General solutions
B908	Any object	в937	$02 \ Constant  ightarrow symb$
B909	Real number	в938	$03 \text{ Function} \rightarrow \text{symb}$
B90A	(Complex num)	В939	14 Payment at end
B90B	"String"	B93A	$19  ightharpoonup V2  ightharpoonup  ext{vector}$
B90C	[ Real array ]	в93в	$20 \text{ Underflow} \rightarrow 0$
B90D	[(Cmpl array)]	B93C	21 Overflow $\rightarrow \pm 9E499$
B90E	{ List }	B93D	$22 \text{ Infinite} \rightarrow \text{error}$
B90F	Name	B93E	27 'X+Y*i' → '(X,Y)'
B910	« Program »	B93F	28 Sequential plot
B911	'Algebraic'	B940	29 Draw axes too
	O		- :=::: =:::=::: = = =

#n	Message	#n	Message
B941	31 Connect points	В970	95 RPN mode
в942	32 Solid cursor	в971	97 List:horiz disp
в943	35 ASCII transfer	в972	98 Vector:horiz disp
В944	36 RECV renames	в973	99 CAS:quiet
в945	37 Single-space prnt	В974	100 Step by step off
В946	38 Add linefeeds	в975	103 Complex off
B947	39 Show I/O messages	B976	105 Exact mode on
B948	40 Don't show clock	B977	106 Simp. in series
B949	41 12-hour clock	в978	109 Sym. factorize
B94A	42 mm/dd/yy format	B979	110 Normal matrices
B94B	43 Reschedule alarm	B97A	111 Simp non rat.
B94C	44 Delete alarm	B97B	112 i simplified
B94D	51 Fraction mark: .	B97C	113 Linear simp on
B94E	52 Show many lines	B97D	114 Disp $1+x \rightarrow x+1$
B94F	53 No extra parens	B97E	115 SQRT simplified
B950	54 Tiny element $\rightarrow 0$	B97F	116 Prefer cos()
B951	55 Save last args	B980	117 CHOOSE boxes
B952	56 Standard beep on	B981	119 Rigorous on
B953	57 Alarm beep on	B982	120 Silent mode off
B954	58 Show INFO	B983	123 Allow Switch Mode
B955	59 Show variables	B984	125 Accur. Sign-Sturm
B956	$60 \ [\alpha] \ [\alpha] \ locks$	B985	126 rref w/ last col
B957	61 [USR][USR] locks	B986	128 Cmplx var allowed
B958	62 User keys off	B987	01 Principal value
B959	63 Custom ENTER off	B988	02 Constant → num
B95A	65 All multiline	B989	$03 \text{ Function} \rightarrow \text{num}$
B95B	66 Stack:x lines str	B98A	14 Payment at begin
B95C	67 Digital clock	B98B	$19 \rightarrow V2 \rightarrow complex$
B95D	68 No AutoIndent	B98C	$20 \text{ Underflow} \rightarrow \text{error}$
B95E	69 Line edit	B98D	$21 \text{ Overflow} \rightarrow \text{error}$
B95F	$70 \rightarrow GROB 1$ line str	B98E	22 Infinite $\rightarrow \pm 9\text{E}499$
B960	71 Show addresses	B98F	27 'X+Y*i' 'X+Y*i'
B961	72 Stack:current fnt	В990	28 Simultaneous plot
B962	73 Edit:current font	B991	29 Don't draw axes
B963	74 Right stack disp	B992	31 Plot points only
B964	75 Key click off	в993	32 Inverse cursor
В965	76 Purge confirm	В994	35 Binary transfer
В966	79 Textbook on	В995	36 RECV overwrites
В967	80 EQW cur stk font	В996	37 Double-space prnt
B968	81 GRB Alg cur font	В997	38 No linefeeds
B969	82 EQW edit cur font	В998	39 No I/O messages
B96A	83 Display grobs on	В999	40 Show clock
B96B	85 Normal stk disp	B99A	41 24-hour clock
B96C	90 CHOOSE:cur font	В99В	42 dd.mm.yy format
B96D	91 MTRW:matrix	B99C	43 Don't reschedule
B96E	92 MASD asm mode	B99D	44 Save alarm
B96F	94 Result = LASTCMD	B99E	51 Fraction mark: ,
2701	O I ITOGOTO - MINI OTILI		or reaction mann,

# <b>n</b>	Message	#n	Message
B99F	52 Show one line	B9CE	114 Disp $x+1 \rightarrow 1+x$
B9A0	53 Show all parens	B9CF	115 SQRT !simplified
B9A1	54 Use tiny element	B9D0	116 Prefer sin()
B9A2	55 No last args	B9D1	117 Soft MENU
B9A3	56 Standard beep off	B9D2	119 Rigorous off
B9A4	57 Alarm beep off	B9D3	120 Silent mode on
B9A5	58 Don't show INFO	B9D4	123 Forb. Switch Mode
B9A6	59 Show names only	B9D5	125 FastSign-no Sturm
B9A7	60 [ $\alpha$ ] locks Alpha	B9D6	126 rref w/o last col
B9A8	61 [USR] locks User	B9D7	128 Vars are reals
B9A9	62 User keys on	B9D8	Object:
B9AA	63 Custom ENTER on	B9D9	Obs in
B9AB	65 Level 1 multiline	B9DA	Name:
B9AC	66 Stk: 1 line str	BA01	1.Send to HP 49
B9AD	67 Analog clock	BA02	2.Get from HP 49
B9AE	68 AutoIndent	BA03	3.Print display
B9AF	69 Infinite line edit	BA04	4.Print
B9B0	70  ightharpoonup GROB x lines str	BA05	5.Transfer
B9B1	71 No addresses	BA06	6.Start Server
B9B2	72 Stack:mini font	BA07	Enter names of vars to send
B9B3	73 Edit:mini font	BA08	Vars in
B9B4	74 Left stack disp	BA09	SEND TO HP 49
B9B5	75 Key click on	BA0A	Port:
B9B6	76 No purge confirm	BA0B	Dbl-Space
в9в7	79 Textbook off	BA0C	Delay:
В9В8	80 EQW mini stk font	BA0D	Xlat:
В9В9	81 GRB Alg mini font	BA0E	Linef
B9BA	82 EQW edit mini fnt	BA0F	Baud:
в9вв	83 Display grobs off	BA10	Parity:
B9BC	85 SysRPL stk disp	BA11	Len:
B9BD	90 CHOOSE:mini font	BA12	Choose print port
B9BE	91 MTRW:list of list	BA13	Enter object(s) to print
B9BF	92 MASD SysRPL mode	BA14	Print extra space between lines?
B9C0	94 Result <> LASTCMD	BA15	Enter delay between lines
B9C1	95 Algebraic mode	BA16	Choose character translations
B9C2	97 List:vert disp	BA17	Print linefeed between lines?
B9C3	98 Vector:vert disp	BA18	Choose baud rate
B9C4	99 CAS:verbose	BA19	Choose parity
B9C5	100 Step by step on	BA1A	Enter printer line length
B9C6	103 Complex on	BA1B	PRINT
B9C7	105 Approx. mode on	BA1C	Type:
B9C8	106 !Simp. in series	BA1D	OvrW
B9C9	109 Num. factorize	BA1E	Fmt:
B9CA	110 Large matrices	BA1F	Chk:
в9СВ	111 !Simp non rat.	BA20	Choose transfer port
B9CC	112 i not simplified	BA21	Choose type of transfer
B9CD	113 Linear simp off	BA22	Enter names of vars to transfer

#n	Message	# <b>n</b>	Message
BA23	Choose transfer format	BB0F	Enter variable column
BA24	Choose checksum type	BB10	Choose statistics type
BA25	Overwrite existing variables?	BB11	Calculate mean?
BA26	TRANSFER	BB12	Calculate standard deviation?
BA27	Local vars	BB13	Calculate variance?
BA28	Remote PC files	BB14	Calculate column total?
BA29	Files in_	BB15	Calculate column maximum?
BA2A	Enter name of dir to change to	BB16	Calculate column minimum?
BA2B	Choose Remote Directory	BB17	Sample
BA2C	Infrared	BB18	Population
BA2D	IR	BB19	FREQUENCIES
BA2E	Wire	BB1A	X-Min:
BA2F	Kermit	BB1B	Bin Count:
BA30	XModem	BB1C	Bin Width:
BA31	Odd	BB1D	Enter minimum first bin X value
BA32	Even	BB1E	Enter number of bins
BA33	Mark	BB1F	Enter bin width
BA34	Space	BB20	FIT DATA
BA35	Spc	BB21	X-Col:
BA36	ASCII	BB22	Y-Col:
BA37	ASC	BB23	Model:
BA38	Binary	BB24	Enter indep column number
BA39	Bin	BB25	Enter dependent column number
BA3A	None	BB26	Choose statistical model
вазв	Newline (Ch 10)	BB27	Correlation
BA3C	Newl	BB28	Covariance
BA3D	Chr 128-159	BB29	PREDICT VALUES
BA3E	ightarrow 159	BB2A	Y:
BA3F	ightarrow 255	вв2в	Enter indep value or press PRED
BA40	Chr 128-255	BB2C	Enter dep value or press PRED
BA41	One-digit arith	BB2D	SUMMARY STATISTICS
BA42	Two-digit arith	BB2E	Calculate:
BA43	Three-digit CRC	BB2F	$\Sigma \mathbf{X}$
BB01	1.Single-var	BB30	$\Sigma \mathbf{Y}$
BB02	2.Frequencies	BB31	$\Sigma$ X2
BB03	3.Fit data	BB32	$\Sigma$ Y2
BB04	4.Summary stats	BB33	$\Sigma XY$
BB05	SINGLE-VARIABLE STATISTICS	BB34	$\mathbf{N}\Sigma$
BB06	$\Sigma$ DAT:	BB35	Calculate sum of X column?
BB07	Type:	BB36	Calculate sum of Y column?
BB08	Mean	BB37	Calculate sum of squares of X?
BB09	Std Dev	BB38	Calculate sum of squares of Y?
BB0A	Variance	BB39	Calculate sum of products?
BB0B	Total	BB3A	Calculate number of data points?
BB0C	Maximum	BB3B	Linear Fit
BB0D	Minimum	BB3C	Logarithmic Fit
BB0E	Enter statistical data	BB3D	Exponential Fit

#n	Message	#n	Message
BB3E	Power Fit	BC2C	12 December
BB3F	Best Fit	BC2D	Week
BB40	5.Hypoth. tests	BC2E	Day
BB41	6.Conf. interval	BC2F	Hour
BC01	1.Browse alarms	BC30	Minute
BC02	2.Set alarm	BC31	Second
BC03	3.Set time, date	BC32	Weeks
BC04	SET ALARM	BC33	Days
BC05	Message:	BC34	Hours
BC06	Time:	BC35	Minutes
BC07	Date:	BC36	Seconds
BC08	Repeat:	BC37	Month/Day/Year
BC09	Enter "message" or « action »	BC38	M/D/Y
BC0A	Enter hour	BC39	Day.Month.Year
BC0B	Enter minute	BC3A	D.M.Y
BC0C	Enter second	BC3B	ALARMS
BC0D	Choose AM, PM, or 24-hour time	BD01	1.Integrate
BC0E	Enter month	BD02	2.Differentiate
BC0F	Enter day	BD03	3. Taylor poly
BC10	Enter year	BD03	4.Isolate var
BC11	Enter alarm repeat multiple	BD04	5.Solve quad
BC12	Enter alarm repeat unit	BD05	6.Manip expr
BC12	SET TIME AND DATE	BD07	INTEGRATE
BC14	Choose date display format	BD07	Expr:
BC14	Monday	BD00	Var:
BC16	Tuesday	BD03	Result:
BC17	Wednesday	BD0A BD0B	Enter expression
BC17	Thursday	BD0B BD0C	Enter variable name
BC19	Friday	BD00	Enter lower limit
BC1A	Saturday	BD0B BD0E	Enter upper limit
BC1B	Sunday	BD0E	Choose result type
BC1C	None	BD01	Choose disp format for accuracy
BC1D	AM	BD10	DIFFERENTIATE
BC1E	PM	BD11	Value:
BC1F	24-hour time	BD12 BD13	Enter variable value
BC20	24-hour time 24-hr	BD13	Expression
BC21	_1 January	BD14	TAYLOR POLYNOMIAL
BC21	_2 February	BD13	Order:
BC23	_3 March	BD10	Enter Taylor polynomial order
	_3 March _4 April	BD17	ISOLATE A VARIABLE
BC24 BC25	_5 May	BD16	Principal
	_6 June	BD19	Get principal solution only?
BC26 BC27	_o June _7 July	BD1A BD1B	SOLVE QUADRATIC
BC27	្នុ <i>t</i> ១៣y ួ8 August	BD1B	MANIPULATE EXPRESSION
BC28 BC29	_9 September	BD1C	MATCH EXPRESSION
	10 October		Pattern:
BC2A		BD1E	Replacement:
BC2B	11 November	BD1F	керіасешені:

#n Message  BD20 Subexpr First  BD21 Cond:  BD22 Enter pattern to search for  BD23 Enter replacement object  BD24 Search subexpressions first?  BD25 Enter conditional expression  BD26 Symbolic  BD27 Numeric  BD27 Numeric  BE20 Enter maximum indep var value  BD27 Plot  BE27 Connect plot points?  BE28 Connect  BE28 W-Tick:  BE28 W-Tick:  BE29 Enter minimum indep var value  BE20 Enter maximum indep var value  BE20 Draw axes before plotting?  BE27 Connect plot points?  BE28 Connect  BE29 Pixels  BE20 Enter minimum indep var value  BE20 Enter maximum indep var value  BE20 Draw axes before plotting?  BE27 Connect plot points?  BE30 Plot functions simultaneously?  BE30 Type:  BE31 Enter indep var increment
BD22 Enter pattern to search for BD23 Enter replacement object BD24 Search subexpressions first? BD25 Enter conditional expression BD26 Symbolic BD27 Numeric BD27 Plot BE20 Enter minimum indep var value BD26 BE20 Enter maximum indep var value BD27 Connect plotting? BE27 Connect plot points? BE30 Plot functions simultaneously?
BD23 Enter replacement object BD24 Search subexpressions first? BD25 Enter conditional expression BD26 Symbolic BD27 Numeric BE01 Plot BB28 V-Tick: BE2C Enter minimum indep var value BE2D Enter maximum indep var value BE2D Draw axes before plotting? BE2F Connect plot points? BE30 Plot functions simultaneously?
BD23 Enter replacement object BD24 Search subexpressions first? BD25 Enter conditional expression BD26 Symbolic BD27 Numeric BE01 Plot BB28 V-Tick: BE2C Enter minimum indep var value BE2D Enter maximum indep var value BE2D Draw axes before plotting? BE2F Connect plot points? BE30 Plot functions simultaneously?
BD24 Search subexpressions first?  BD25 Enter conditional expression  BD26 Symbolic  BD27 Numeric  BE01 Plot  BE20 Enter minimum indep var value  BE2D Enter maximum indep var value  BE2E Draw axes before plotting?  BE2F Connect plot points?  BE30 Plot functions simultaneously?
BD25Enter conditional expressionBE2DEnter maximum indep var valueBD26SymbolicBE2EDraw axes before plotting?BD27NumericBE2FConnect plot points?BE01PlotBE30Plot functions simultaneously?
BD26SymbolicBE2EDraw axes before plotting?BD27NumericBE2FConnect plot points?BE01PlotBE30Plot functions simultaneously?
BD27 Numeric BE2F Connect plot points? BE01 Plot BE30 Plot functions simultaneously?
BE01 Plot BE30 Plot functions simultaneously?
·
DEST THE HIGH THE CHICHE
BE03 \( \alpha:\) BE32 Indep step units are pixels?
BE04 H-View: BE33 Enter horizontal tick spacing
BE05 Autoscale BE34 Enter vertical tick spacing
BE06 V-View: BE35 Tick spacing units are pixels?
BE07 Choose type of plot BE36 Depnd:
BE08 Choose angle measure BE37 Enter dependent var name
BE09 Enter function(s) to plot BE38 Enter minimum dep var value
BEOA Enter minimum horizontal value BE39 Enter maximum dep var value
BEOB Enter maximum horizontal value BE3A H-Var:
BEOC Autoscale vertical plot range? BE3B V-Var:
BEOD Enter minimum vertical value BE3C Enter max indep var increment
BEOE Enter maximum vertical value BE3D Choose horizontal variable
BEOF Plot $(x(t), y(t))$ BE3E Choose vertical variable
BE10 Enter complex-valued func(s)  BE3F 0 INDEP
BE11 Plot $y'(t)=f(t,y)$ BE40 1 SOLN
BE12 Enter function of INDEP and SOLN BE41 SOLN(
BE13 Enter derivative w.r.t. SOLN BE42 X-Left:
BE14 Enter derivative w.r.t. INDEP BE43 X-Right:
BE15 Use Stiff diff eq solver? BE44 Y-Near:
BE16 $\Sigma$ Dat: BE45 Y-Far:
BE17 Col: BE46 Z-Low:
BE18 Wid: BE47 Z-High:
BE19 Enter data to plot BE48 Enter minimum X view-volume val
BE1A Arrays in BE49 Enter maximum X view-volume val
BE1B Enter column to plot BE4A Enter minimum Y view-volume val
BE1C Enter bar width BE4B Enter maximum Y view-volume val
BE1D Cols: BE4C Enter minimum Z view-volume val
BE1E Enter col to use for horizontal BE4D Enter maximum Z view-volume val
BE1F Enter col to use for vertical BE4E XE:
BE20 Steps: BE4F YE:
BE21 Enter indep var sample count BE50 ZE:
BE22 Enter dep var sample count BE51 Enter X eyepoint coordinate
BE23 Plot Options BE52 Enter Y eyepoint coordinate
BE24 Lo: BE53 Enter Z eyepoint coordinate
BE25 Hi: BE54 Save Animation
BE26 Axes BE55 Save animation data after plot?
BE27 Simult BE56 XX-Left:

# <b>n</b>	Message	#n	Message
BE57	XX-Rght:	BF0F	f:
BE58	YY-Near:	BF10	$\partial \mathbf{f} \partial \mathbf{y}$ :
BE59	YY-Far:	BF11	$\partial \mathbf{f} \partial \mathbf{t}$ :
BE5A	Enter minimum XX range value	BF12	Indep:
BE5B	Enter maximum XX range value	BF13	Init:
BE5C	Enter minimum YY range value	BF14	Final:
BE5D	Enter maximum YY range value	BF15	Soln:
BE5E	XX and YY Plot Options	BF16	Tol:
BE5F	Zoom Factors	BF17	Step:
BE60	H-Factor:	BF18	Stiff
BE61	V-Factor:	BF19	Enter function of INDEP and SOLN
BE62	Recenter at Crosshairs	BF1A	Enter derivative w.r.t. SOLN
BE63	Enter horizontal zoom factor	BF1B	Enter derivative w.r.t. INDEP
BE64	Enter vertical zoom factor	BF1C	Enter independent var name
BE65	Recenter plot at crosshairs?	BF1D	Enter initial indep var value
BE66	Reset plot	BF1E	Enter final indep var value
BE67	Dflt	BF1F	Enter solution var name
BE68	Auto	BF20	Enter initial solution var value
BE69	Function	BF21	Press SOLVE for final soln value
BE6A	Polar	BF22	Enter absolute error tolerance
BE6B	Conic	BF23	Enter initial step size
BE6C	Truth	BF24	Calculate stiff differential?
BE6D	Parametric	BF26	Tolerance
BE6E	Diff Eq	BF27	Solution
BE6F	Histogram	BF28	SOLVE $AN \cdot X^N + + A1 \cdot X + A0$
BE70	Bar	BF29	Coefficients [ an a1 a0 ]:
BE71	Scatter	BF2A	Roots:
BE72	Slopefield	BF2B	Enter coefficients or press SOLVE
BE73	Wireframe	BF2C	Enter roots or press SOLVE
BE74	Ps-Contour	BF2D	Coefficients
BE75	Y-Slice	BF2E	Roots
BE76	Gridmap	BF2F	SOLVE SYSTEM A:X=B
BE77	Pr-Surface	BF30	A:
BF01	1. Solve equation	BF31	B:
BF02	2.Solve diff eq	BF32	X:
BF03	3. Solve poly	BF33	Enter coefficients matrix A
BF04	4.Solve lin sys	BF34	Enter constants or press SOLVE
BF05	5. Solve finance	BF35	Enter solutions or press SOLVE
BF06	SOLVE EQUATION	BF36	Constants
BF07	Enter value or press SOLVE	BF37	Solutions
BF08	Eq:	BF38	N:
BF09	Enter function to solve	BF39	I%YR:
BF0A	Funcs in	BF3A	PV:
BF0B	Solver Variable Order	BF3B	PMT:
BF0C	Variables:	BF3C	P/YR:
BF0D	Enter order of vars to display	BF3D	FV:
BF0E	SOLVE $Y'(T)=F(T,Y)$	BF3E	Enter no. of payments or SOLVE
	·	1	* *

#n	Message	#n	Message
BF3F	Enter yearly int rate or SOLVE	DE18	Pivots
BF40	Enter present value or SOLVE	DE19	Press CONT to go on
BF41	Enter payment amount or SOLVE	DE1A	$\mathrm{Test}_{\sqcup}$
BF42	Enter no. of payments per year	DE1B	To be implemented
BF43	Enter future value or SOLVE	DE1C	Unable to factor
BF44	Choose when payments are made	DE1D	$Z \text{ is not} = 1 \mod 4$
BF45	TIME VALUE OF MONEY	DE1E	Z is not prime
BF47	I%/YR	DE1F	Empty {} of equations
BF48	PV	DE20	Not reducible to a rational expression
BF49	PMT	DE21	Non unary operator
BF4A	FV	DE22	User function
BF4B	End	DE23	Non isolable operator
BF4C	Begin	DE24	Not exact system
BF4D	Beg	DE25	Parameters not allowed
BF4E	AMORTIZE	DE26	CAS internal error
BF4F	Payments:	DE27	Invalid ^ for SERIES
BF50	Principal:	DE28	Operator not implemented (SERIES)
BF51	Interest:	DE29	No variable in expr.
BF52	Balance:	DE2A	No solution found
BF53	Enter no. of payments to amort	DE2B	Invalid derivation arg
BF54	Principal	DE2C	No solution in ring
BF55	Interest	DE2D	Not a linear system
BF56	Balance	DE2E	Can't derive int. var
C001	Unable to find root	DE2F	Diff equation order>2
DE01	_denominator(s)_	DE30	INT:invalid var change
DE02	root(s)	DE31	Mode switch cancelled
DE03	$\operatorname{last}_{\lrcorner}$	DE32	No name in expression
DE04	$obvious$ _	DE33	Invalid user function
DE05	factorizing_	DE34	Can't find ODE type
DE06	value_	DE35	Integer too large
DE07	$\_{ m test}({ m s})_{\_}$	DE36	Unable to find sign
DE08	searching_	DE37	Non-symmetric matrix
DE09	TAYLR of $\downarrow$ at $\Box$	DE38	ATAN insufficent order
DE0A	$\mathrm{nth}_{oldsymbol{oldsymbol{\sqcup}}}$	DE39	ASIN at infinity undef
DE0B	_is_	DE3A	Unsigned inf error
DE0C	$\_$ numerator(s) $\_$	DE3B	LN[Var] comparison err
DE0D	Less than_	DE3C	Undef limit for var
DE0E	$\operatorname{multiplicity}_{\square}$	DE3D	Bounded var error
DEOF	list of_	DE3E	Got expr. indep of var
DE10	_at_	DE3F	Can't state remainder
DE11	factor(s)	DE40	LN of neg argument
DE12	Eigenvalues_	DE41	Insufficient order
DE13	Computing for	DE42	ABS of non-signed 0
DE14	Root mult <	DE43	Numeric input
DE15	Numerical to symbolic	DE44	Singularity! Continue?
DE16	Invalid operator	DE45	Cancelled
DE17	Result:	DE46	Negative integer

#n	Message	#n	Message
DE47	Parameter is cur. var. dependent	E10E	Dirac's
DE48	Unsimplified sqrt	E10F	electronic charge
DE49	Non polynomial system	E110	electron mass
DE4A	Unable to solve ODE	E111	g/me ratio
DE4B	Array dimension too large	E112	proton mass
DE4C	Unable to reduce system	E113	mp/me ratio
DE4D	Complex number not allowed	E114	fine structure
DE4E	Polyn. valuation must be 0	E115	mag flux quantum
DE4F	Mode switch not allowed here	E116	Faraday
DE50	Non algebraic in expression	E117	Rydberg
DE51	Purge current variable_	E118	Bohr radius
DE52	Reduction result	E119	Bohr magneton
DE53	Matrix not diagonalizable	E11A	nuclear magneton
DE54	Int[u'*F(u)] with $u=$	E11B	photon wavelength
DE55	Int. by part u'*v, u=	E11C	photon frequency
DE56	Square root_	E11D	Compton wavelen
DE57	Rational fraction	E11E	1 radian
DE58	Linearizing	E11F	$2\pi$ radians
DE59	Risch alg. of tower	E120	∠ in trig mode
DE5A	Trig. fraction, u=	E121	Wien's
DE5B	Unknown operator (DOMAIN)	E122	k/q
DE5C	Same points	E123	4 ε0/q
DE5D	Unsigned inf. Solve?	E124	$q^* \varepsilon 0$
DE5E	CAS not available	E125	dielectric const
DE5F	Can not store current var	E126	SiO2 dielec cons
DE60	Not available on the HP40G	E127	ref intensity
DE61	Not available on the HP49G	E128	CONSTANTS LIBRARY
DE62	SERIES remainder is O(1) at order 3	E129	Undefined Constant
DE63	Delta/Heaviside not available from HOME	E401	Invalid Mpar
DE64	Warning, integrating in approx mode	E402	Single Equation
DE65	Function is constant	E403	EQ Invalid for MINIT
DE66	Can not unbind local vars	E404	Too Many Unknowns
DE67	Replacing strict with large inequality	E405	All Variables Known
DE68	No valid environment stored	E406	Illegal During MROOT
E101	Avogadro's number	E407	Solving for
E102	Boltzmann	E408	Searching
E103	molar volume	E601	No Solution
E104	universal gas	E602	Many or No Solutions
E105	std temperature	E603	I%YR/PYR < -100
E106	std pressure	E604	Invalid N
E107	Stefan-Boltzmann	E605	Invalid PYR
E108	speed of light	E606	Invalid #Periods
E109	permittivity	E607	Undefined TVM Variable
E10A	permeability	E608	END mode
E10B	accel of gravity	E609	BEGIN mode
E10C	gravitation	E60A	_payments/year
E10D	Planck's	E60B	Principal
	<del></del>		~-r

# <b>n</b>	Message	#n	Message
E60C	Interest	10105	Val betw 0-15 expected
E60D	Balance	10106	Val betw 1-16 expected
E701	$NEAR_{\perp}$	10107	Label Expected
E702	$\_{MINE}$	10108	Hexa Expected
E703	_MINES	10109	Decimal Expected
E704	SCORE:_	1010A	Can't Find
E705	YOU MADE IT!!	1010B	Label already defined
E706	YOU BLEW UP!!	1010C	{ expected
10001	Invalid \$ROMID	1010D	} expected
10002	Invalid \$TITLE	1010E	( expected
10003	Invalid \$MESSAGE	1010F	Forbidden
10004	Invalid \$VISIBLE	10110	Bad Expression
10005	Invalid \$HIDDEN	10111	Jump too Long
10006	Invalid \$EXTPRG	10112	Val betw 1-8 expected
10101	Invalid File	10113	Insuffisant Memory
10102	Too Many	10114	Matrix Error
10103	Unknown Instruction	10115	Define Error
10104	Invalid Field	31401	No Message here

## Part VI

## Index

## **Appendix F Entries sorted by Name**

The entries in this index have been sorted alphabetically, ignoring case. Leading characters  $\hat{\ }$ ,  $\hat{\ }$ , and x have no influence on the position of an entry. Entries starting with a digit or a symbol are at the end of the index. Note that for technical reasons, the page number given may be off by one for a few percent of the entries. If the page reference is 167, the entry may actually be the first entry on page 168.

Addr.	Name	Page	Addr.	Name	Page
25F0F	a%>\$	46	378006	^ADDMATOBJext	345
25F0F	a%>\$,	46	295006	^ADDMULTIPL	384
106007	^A/B2PQR	425	112007	^ADDONEVAR	379
003100	$x \mathbb{A} {\longrightarrow}$	478	2B7CC	addrClkOnNib	
004100	$xA \rightarrow H$	478	00A0E	addrKEYSTATE	
20B006	^ABCUV	414	01661	addrORghost	
030314	~xABCUV	453	04E66	addrTEMPENV	
07497	ABND	117	2ACA9	addrTEMPTOP	
04EA4	ABORT	156	13D006	^addt!=	363
39A07	xABS	453	13F006	^addt%	363
50D006	^xABSext	348	141006	^addt%CH	363
25FA4	ABUFF	272	143006	^addt%T	363
390E4	xACK	453	38B006	^addt/	359
2F319	ACK_INIT		406006	^addt0meta	77
390C9	xACKALL	453	38A006	^addt2	359
3A7DC	xACOS	453	133006	^addt<	363
43C006	^ACOS2ASIN	352	135006	^addt<=	363
425006	^acos2ln	366	13B006	^addt==	363
424006	^ACOS2LN	351	137006	^addt>	363
025314	~xACOS2S	453	139006	^addt>=	363
43A006	^ACOS2Sext	357	398006	^addt^	360
533006	^xACOSext	349	50E006	^addtABS	362
3A8D8	xACOSH	453	510006	^addtABSEXACT	362
451006	^acosh2ln	367	53E006	^addtACOS	364
450006	^ACOSH2LNext	352	54F006	^addtACOSH	364
54E006	^xACOSHext	349	56E006	^addtALOG	365
315BB	ADDF		151006	^addtAND	363
2EF76	AddLeadingSpace	312	513006	^addtARG	362
279006	^ADDLISText	381	53C006	^addtASIN	364

Addr.	Name	Page	Addr.	Name	Page
54C006	^addtASINH	364	2F179	AdjEdModes	
540006	^addtATAN	364	047CF	adrDISABLE_K	
549006	^addtATANH	364	047DD	adrKEYBUFFER	
560006	^addtCEIL	365	2AF37	AEQ1stcase	143
14B006	^addtCOMB	363	2B06A	AEQopscase	143
506006	^addtCONJ	363	071AB	AGAIN	151
535006	^addtCOS	363	31123	aH>HMS	
544006	^addtCOSH	364	25FA9	ALARM?	174
55A006	^addtD->R	364	2F178	ALARMS@	174
558006	^addtEXP	364	25E7A	ALARMxcp	
570006	^addtEXPM	365	38093	xALG->	
574006	^addtFACT	365	0A8006	^ALG48FCTR?	374
55E006	^addtFLOOR	365	093006	^ALG48MSOLV	384
564006	^addtFP	365	080314	xALGB	454
39E006	^addti	360	11A007	^ALGCASCOMPEVAL	353
2619D	addtics		2BE36	ALGcase	146
556006	^addtINV	364	2F2DA	AlgCharEdit	
562006	^addtIP	365	2AA43	AlgDecomp	53
523006	^addtLN	363	256EA	AlgEntry?	278
56A006	^addtLNP1	365	25E7B	ALGeq?	
56C006	^addtLOG	365	2F1AF	AlgObEdit	311
568006	^addtMANT	365	00E004	^algparse	
131006	^addtMAX	362	00F004	^algunwrap	
12F006	^addtMIN	362	000FF	allkeys	
241006	^ADDTMOD	416	2F177	AllowPrlcdCl	
145006	^addtMOD	363	3AAE5	xALOG	453
06E314	~xADDTMOD	453	41C006	^ALOG2EXP	351
577006	^addtNOT	365	31066	aMODF	
14F006	^addtOR	363	04B0AB	xAMORT	453
0C9007	^ADDTOREAL	408	3CA07	xAND	453
0000DE	xADDTOREAL	453	03B46	AND	136
14D006	^addtPERM	363	25E7C	AND\$	49
55C006	^addtR->D	365	36D4E	ANDcase	139
2EF75	AddTrailingSpace	312	36EED	ANDITE	139
149006	^addtRND	363	36E6B	ANDNOTcase	139
511006	^addtSIGN	362	3F033	xANS	453
537006	^addtSIN	364	33107	any	10
53B006	^addtSINACOS	364	012100	XAPEEK	478
542006	^addtSINH	364	2F31A	APNDCRLF	47, 181
554006	^addtSQ	364	2EF5A	apndvarlst	73
551006	^addtSQRT	364	29F25	AppDisplay!	223
539006	^addtTAN	364	29F35	AppDisplay@	223
410006	^addtTAN/2	366	35BD7	APPEND_SPACE	49
546006	^addtTANH	364	0FE006	^AppendList	73
147006	^addtTRNC	363	2A145	AppError!	223
153006	^addtXOR	363	2A158	AppError@	223
566006	^addtXPON	365	2A055	AppExitCond!	223
12D006	^addtXROOT	362	2A065	AppExitCond@	223

Addr.	Name	Page	Addr.	Name	Page
29F55	AppKeys!	223	43E006	^atan2ln	366
29F75	AppKeys0	223	43D006	^ATAN2LNext	352
3D7AC	xAPPLY	453	022314	~xATAN2S	454
25690	AppMode?	224	430006	^ATAN2Sext	356
2EEEE	APPprompt1!		534006	^xATANext	349
2F17A	APPprompt2		3A94F	xATANH	454
07B007	^Approx	419	453006	^atanh2ln	367
46C006	^APPROXCOMPEVAL	353	452006	^ATANH2LNext	352
068004	^Arbo		548006	^xATANHext	350
3C8C6	xARC	453	3EB64	xATTACH	454
3EAC7	xARCHIVE	453	27882	Attn#	19
3A390	xARG	453	25FAE	ATTN?	207
515006	^ARG2	38	338C3	ATTNERR	19
085314	~xARIT	453	05040	ATTNFLG@	207
17E006	^ARRAY2MATRIX	338	05068	ATTNFLGCLR	207
3312F	arry	10	25E7D	ATTNxcp	
3BEC5	xARRY>	454	2EF6C	AtUserStack	197
3384B	ARRYLISTREAL	18	0130DE	xAUGMENT	
33391	ARRYREAL	13	3C49F	xAUTO	454
3382D	ARRYREALREAL	17	2EEEF	AUTOSCALE	318
003007	^ArryToList	338	3C3B2	xAXES	454
002007	^ArryToMatrix	65	04A314	~xAXL	454
35E006	^ARSIZE	64	049314	~xAXM	454
3A756	xASIN	454	382006	^AXQ	345
437006	^ASIN2ACOS	352	04C314	~xAXQ	454
434006	^ASIN2ATAN	352	38F21	xB>R	454
435006	^asin2atan	366	33661	backup	
024314	~xASIN2C	454	0905F	BAK>OB	101
436006	^ASIN2Cext	357	081D9	BAKNAME	101
423006	^asin2ln	366	35C006	^BANGARRY	343
422006	^ASIN2LN	351	3C9D3	xBAR	454
023314	~xASIN2T	454	3E196	xBARPLOT	454
433006	^ASIN2Text	356	080314	~xBASE	454
532006	^xASINext	349	2EFBF	BASE	177
3A88E	xASINH	454	2F9006	^base_ln	420
44F006	^asinh2ln	367	0110DE	xBASIS	
44E006	^ASINH2LNext	352	3EDCC	xBAUD	454
54B006	^xASINHext	350	0530B3	~BBDownArrow	246
2F1A5	AskQuestion	282	05B0B3	~BBEmpty?	246
01E100	xASM	479	05C0B3	$^{\sim}$ BBGetDefltHeight	246
018100	$xASM \rightarrow$	479	03F0B3	~BBGetN	246
3EEE7	xASN	454	0370B3	~BBGetNGrob	245
2F3B3	AsnKey	208	0380B3	~BBGetNStr	245
38DE1	xASR	454	04B0B3	~BBIsChecked?	246
0260DE	xASSUME		0150B3	~BBMoveTo	245
3A844	×ATAN	454	0590B3	~BBPgDown	246
432006	^atan2asin	366	05A0B3	~BBPgUp	246
431006	^ATAN2ASIN	352	0190B3	~BBRecalOff&Disp	245
			1	-	

Addr.	Name	Page	Addr.	Name	Page
02F0B3	~BBReDrawBackgr	245	33567	BINT112	15
03B0B3	~BBRereadChkEnbl	246	33571	BINT113	15
0250B3	~BBReReadCoords	245	3357B	BINT114	15
03C0B3	~BBRereadFullScr	246	33585	BINT115	15
0240B3	~BBReReadHeight	245	3358F	BINT116	15
03E0B3	~BBReReadNElems	246	33599	BINT117	15
0230B3	~BBReReadPageSize	245	335A3	BINT118	15
0260B3	~BBReReadWidth	245	335AD	BINT119	15
0290B3	~BBRunCanclAction	245	3317F	BINT12	11
0280B3	~BBRunENTERAction	245	335B7	BINT120	15
0220B3	~BBRunEntryProc	245	335C1	BINT121	15
0540B3	~BBSpace	246	335CB	BINT122	15
0520B3	~BBUpArrow	246	335D5	BINT123	15
39765	xBEEP	454	335DF	BINT124	15
071A2	BEGIN	150	335E9	BINT125	15
08F006	^Berlekamp	373	335F3	BINT126	16
08E006	^BerlekampP	373	335FD	BINT127	16
3EA006	^BESTDIV2	357	33607	BINT128	16
3E2C1	xBESTFIT	454	33611	BINT129	16
169006	^BESTMATRIXTYPE	65	33189	BINT13	11
019100	xBetaTesting	479	3361B	BINT130	16
4C8006	^BEZOUTMSOLV	396	3361B	BINT130d	16
0CF006	^BFactor	331	33625	BINT131	16
280006	^BICARREE?	382	33625	BINT131d	16
25FB3	BIGDISPN	281	33193	BINT14	11
25FB8	BIGDISPROW1	281	3319D	BINT15	11
25FBD	BIGDISPROW2	281	331A7	BINT16	11
25FC2	BIGDISPROW3	281	331B1	BINT17	11
25FC7	BIGDISPROW4	281	331BB	BINT18	11
3B655	xBIN	454	331C5	BINT19	11
074D0	BIND	116	3311B	BINT2	10
3E171	xBINS	454	331CF	BINT20	11
33107	BINTO	10	331D9	BINT21	11
33111	BINT1	10	331E3	BINT22	11
3316B	BINT10	10	331ED	BINT23	11
334EF	BINT100	15	331F7	BINT24	11
334F9	BINT101	15	33201	BINT25	11
33503	BINT102	15	3371F	BINT253	16
3350D	BINT103	15	33729	BINT255d	17
33517	BINT104	15	3320B	BINT26	11
33521	BINT105	15	33215	BINT27	11
3352B	BINT106	15	3321F	BINT28	11
33535	BINT107	15	33229	BINT29	11
3353F	BINT108	15	33125	BINT3	10
33549	BINT109	15	33233	BINT30	12
33175	BINT11	10	3323D	BINT31	12
33553	BINT110	15	33247	BINT32	12
3355D	BINT111	15	33251	BINT33	12

Addr.	Name	Page	Addr.	Name	Page
3325B	BINT34	12	33409	BINT77	14
33265	BINT35	12	33413	BINT78	14
3326F	BINT36	12	3341D	BINT79	14
33279	BINT37	12	33157	BINT8	10
33283	BINT38	12	33427	BINT80	14
3328D	BINT39	12	33607	BINT80h	16
3312F	BINT4	10	33431	BINT81	14
33297	BINT40	12	3343B	BINT82	14
33387	BINT40h	13	33445	BINT83	14
332A1	BINT41	12	3344F	BINT84	14
332AB	BINT42	12	33459	BINT85	14
332B5	BINT43	12	33463	BINT86	14
332BF	BINT44	12	3346D	BINT87	14
332C9	BINT45	12	33477	BINT88	14
332D3	BINT46	12	33481	BINT89	14
332DD	BINT47	12	33161	BINT9	10
332E7	BINT48	12	3348B	BINT90	14
332F1	BINT49	12	33495	BINT91	14
33139	BINT5	10	3349F	BINT92	14
332FB	BINT50	13	334A9	BINT93	14
33305	BINT51	13	334B3	BINT94	15
3330F	BINT52	13	334BD	BINT95	15
33319	BINT53	13	334C7	BINT96	15
33323	BINT54	13	334D1	BINT97	15
3332D	BINT55	13	334DB	BINT98	15
33337	BINT56	13	334E5	BINT99	15
33341	BINT57	13	33585	BINT_115d	15
3334B	BINT58	13	3358F	BINT_116d	15
33355	BINT59	13	335CB	BINT_122d	15
33143	BINT6	10	3361B	BINT_130d	16
3335F	BINT60	13	33625	BINT_131d	16
33369	BINT61	13	33751	BINT_263d	17
33373	BINT62	13	33495	BINT_91d	14
3337D	BINT63	13	334C7	BINT_96d	15
33387	BINT64	13	336D9	BINTC0h	16
33391	BINT65	13	0D4006	^BIsPrime?	332
3339B	BINT66	13	2EFC5	bit#%*	57
333A5	BINT67	13	2EFC9	bit#%+	57
333AF	BINT68	14	2EFC7	bit#%-	57
333B9	BINT69	14	2EFC3	bit#%/	58
3314D	BINT7	10	2EFC4	bit%#*	57
333C3	BINT70	14	2EFC8	bit%#+	57
333CD	BINT71	14	2EFC6	bit%#-	57
333D7	BINT72	14	2EFC2	bit%#/	57
333E1	BINT73	14	2EFBC	bit*	57
333EB	BINT74	14	2EFB9	bit+	57
333F5	BINT75	14	2EFBA	bit-	57
333FF	BINT76	14	2EFBD	bit/	57

Addr.	Name	Page	Addr.	Name	Page
2EFAC	bitAND	58	27DBF	C%-1	36
2EFB8	bitASR	58	2B2A7	C%-1=case	143
2EFAF	bitNOT	58	27DE4	C%0	36
2EFAD	bitOR	58	261E8	C%0=	38
2EFB6	bitRL	58	2B15D	C%0=case	142
2EFB7	bitRLB	58	27E09	C%1	36
2EFB4	bitRR	58	25E81	C%1/	37
2EFB5	bitRRB	58	2B1C1	C%1=case	142
2EFB0	bitSL	58	2B22A	C%2=case	143
2EFB1	bitSLB	58	2F31F	C%>#	22
2EFB2	bitSR	58	05D2C	C%>%	31
2EFB3	bitSRB	58	25E82	C%>%%	37
2EFAE	bitXOR	58	25E83	C%>%%SWAP	37
3C70A	xBLANK	454	18B006	^C%>C%%	37
2F16D	Blank\$	48	25E84	C%ABS	37
2EF5E	BlankDA1	277	25E85	C%ACOS	38
2EE5C	BlankDA12	277	25E86	C%ACOSH	38
2F31B	BlankDA2	277	25E87	C%ALOG	38
2F31C	BlankDA2a	277	25E88	C%ARG	38
25E7E	BLANKIT	277	25E89	C%ASIN	38
2F31D	BOTROW	279	25E8A	C%ASINH	38
3C6E0	xBOX	454	25E8B	C%ATAN	38
25E7F	Box/StdLabel	95	25E8C	C%ATANH	38
25E80	Box/StdLbl:	95	25E8F	C%C^C	37
0D5006	^BRabin	332	25E90	C%C^R	37
0100E0	~BRbrowse		261ED	C%CHS	37
0A5003	^BRDispItems		261F2	C%CONJ	37
0A4003	^BRdone		25E8D	C%COS	38
0D0006	^BrentPow	331	25E8E	C%COSH	38
03D0B3	~BReReadMenus	246	25E91	C%EXP	38
0AB003	^BRGetItem		25E92	C%LN	38
0A6003	^BRinverse		25E93	C%LOG	38
0130E0	~BRoutput		25E94	C%R^C	37
070004	^BrowseMem.1		25E95	C%SGN	37
0190E0	~BRRclC1	246	25E96	C%SIN	38
0180E0	~BRRclCurRow		25E97	C%SINH	38
0030E0	~BRStoC1		25E98	C%SQRT	37
0A7003	^BRViewItem		25E99	C%TAN	38
3EE47	xBUFLEN	454	25E9A	C%TANH	38
2F31E	BUILDKPACKET		188006	^C2C%%	37
39480	xBYTES	455	01E0DE	xC2P	455
261D4	C%%0=	38	25E9B	C>Im%	37
27E2E	C%%1	36	3C58E	xC>PX	456
05DBC	C%%>%%	37	3BAF5	xC>R	456
261D9	C%%>C%	37	25E9C	C>Re%	37
261DE	C%%CHS	38	34D00	CACHE	117
261E3	C%%CONJ	38	2EF72	CacheStack	
51E006	^C%%SQRT	38	2EF91	CAL_CURS_POS	301

bitAND - CHR\_: 505

Addr.	Name	Page	Addr.	Name	Page
2EF90	CAL_CURS_POS_VIS	301	2F320	CHECKHEIGHT	90
2B2F2	CallEditCmd:	311	04708	CHECKKEY	205
57F006	^CANTFACTOR	403	25F2B	CHECKMENU	293
050ED	CAR\$	47	25EA6	CheckMenuRow	293
05089	CARCOMP	68	2F162	CHECKPICT	316
15B006	^CARCOMPext	372	0AA006	^CheckPNoExt	374
07E314	~xCASCFG	455	2F163	CHECKPVARS	316
0330DE	xCASCMD	455	07D007	^CHECKSING	402
467006	^CASCOMPEVAL	353	03A314	~xCHINREM	455
46B006	^CASCRUNCH	353	00B0DE	xCHOLESKY	
349F9	case	139	04D0AB	xCHOOSE	455
38B28	xCASE	455	0000B3	~Choose	244
349D6	case2DROP	140	070002	^Choose2	233
34985	case2drop	139	072002	^Choose3	233
365CC	case2drpfls	140	076002	^Choose3CANCL	234
36C4F	caseDEADKEY	146	074002	^Choose3Index	233
36C4F	caseDoBadKey	146	077002	^Choose30K	234
349B1	caseDROP	139	073002	^Choose3Save	233
3495D	casedrop	139	2F15A	CHOOSE_INIT	
36C36	caseDrpBadKy	146	075002	^ChooseDefHandler	233
365B3	casedrpfls	139	0050B3	~ChooseMenu0	244
368FB	casedrptru	139	0060B3	~ChooseMenu1	244
361B2	caseERRJMP	146	0070B3	~ChooseMenu2	$\frac{244}{244}$
365E5	caseFALSE	140	0630B3	~ChooseSimple	244
36B53	caseSIZEERR	146	3BC19	xCHR	455
3652C	caseTRUE	140	05A51	CHR>#	$\frac{100}{22}$
2AA70	CASEVAL		37AA5	CHR>\$	47
1D5006	^CASFLAGEVAL	408	33D40	CHR_#	41
466006	^CASNUMEVAL	353	33D47	CHR_*	41
007100	xCD→	478	33D17	CHR_+	41
3EB006	^CDIV2ext	110	33D1E	CHR_,	41
0516C	CDR\$	47	33D5C	CHR	41
05153	CDRCOMP	68	33F46	CHR>	43
3AD1B	xCEIL	455	33D63	CHR	41
25FEF	CENTER\$3x5	96	33D33	CHR	41
25FEF 25FF4	CENTER\$5x7	97	33D52	CHR_/	41
3C3DC	xCENTR	455	33D0A 33D71	CHR_0	41
3B4E9	xCF	455	33D71 33D2B	CHR_00	41
2DA2B	cfC	30	33D2B 33D78	CHR_00	41
2DA2B 2DA11		30	33D76 33D7F	CHR_2	41
08F007		405	33D7F 33D86	CHR_2 CHR_3	41
2B5006	^CFGDISPLAY ^CGCDext				41
	^ChangeFocus	333	33D8D	CHR_4	
0BE002	9	170	33D94	CHR_5	$\begin{array}{c} 41 \\ 41 \end{array}$
05AB3	CHANGETYPE	179 15	33D9B	CHR_6	
3355D	char	19	33DA2	CHR_7	41
2F1AD	CharEdit	50	33DA9	CHR_8	41
2F1A7	CHARSEDIT	50	33DB0	CHR_9	41
26210	CHECK_SCAN_FONT		33DB7	CHR_:	41

Addr.	Name	Page	Addr.	Name	Page
33DBE	CHR_;	41	33EF9	CHR_p	43
33DC5	CHR_<	41	33F7E	CHR_Pi	43
33F4D	CHR_<<	43	33E4A	CHR_Q	42
33FBD	CHR_<=	43	33F00	CHR_q	43
33FCB	CHR_<>	43	33F07	CHR_r	43
33DCC	CHR_=	41	33E51	CHR_R	42
33DD3	CHR_>	41	33F85	CHR_RightPar	41
33FC4	CHR_>=	43	33F0E	CHR_s	43
33F54	CHR_>>	43	33E58	CHR_S	42
33FA1	CHR_[	42	33F8C	CHR_Sigma	43
33FA8	CHR_]	42	33F93	CHR_Space	41
33E90	CHR_a	42	33F15	CHR_t	43
33DDA	CHR_A	41	33E5F	CHR_T	42
33F5B	CHR_Angle	43	33E66	CHR_U	42
33DE1	CHR_B	41	33F1C	CHR_u	43
33E97	CHR_b	42	33F9A	CHR_UndScore	42
33DE8	CHR_C	41	33E6D	CHR_V	42
33E9E	CHR_c	42	33F23	CHR_v	43
33EA5	CHR_d	42	33F2A	CHR_w	43
33DEF	CHR_D	42	33E74	CHR_W	42
33D39	CHR_DblQuote	41	33F31	CHR_x	43
33F62	CHR_Deriv	43	33E7B	CHR_X	42
33EAC	CHR_e	42	33E82	CHR_Y	42
33DF6	CHR_E	42	33F38	CHR_y	43
33EB3	CHR_f	42	33E89	CHR_Z	42
33DFD	CHR_F	42	33F3F	CHR_z	43
33E04	CHR_G	42	33FAF	CHR_{	43
33EBA	CHR_g	42	33FB6	CHR_}	43
33EC1	CHR_h	42	01D0DE	xCIRC	455
33E0B	CHR_H	42	2F215	CircleB	93
33E12	CHR_I	42	2F216	CircleG1	93
33EC8	CHR_i	42	2F217	CircleG2	93
33F69	CHR_Integral	43	2F218	CircleW	93
33E19	CHR_J	42	2F219	CircleXor	93
33ECF	CHR_j	42	2F13D	CK#-	23
33ED6	CHR_k	42	51D006	^CK%%SQRT	34
33E20	CHR_K	42	203006	^CK&CONV2INT	328
33EDD	CHR_1	42	202006	^CK&CONVINT	328
33E27	CHR_L	42	25EA7	Ck&DecKeyLoc	205
33F70	CHR_LeftPar	41	2631E	CK&DISPATCH0	197
33E2E	CHR_M	42	26328	CK&DISPATCH1	197
33EE4	CHR_m	43	26323	CK&DISPATCH2	197
33EEB	CHR_n	43	007002	^Ck&DoMsgBox	283
33E35	CHR_N	42	25EA8	Ck&Freeze	
33F77	CHR_Newline	41	262B0	CK0	196
33E3C	CHR_O	42	25E9D	CK0ATTNABORT	207
33EF2	CHR_o	43	26292	CK0NOLASTWD	196
33E43	CHR_P	42	262B5	CK1	196

Addr.	Name	Page	Addr.	Name	Page
26300	CK1&Dispatch	197	3DD4E	xCLEAR	455
187006	^CK1Cext	201,333	261C5	CLEARLCD	277
25E9E	CK1NoBlame	197	2F15B	CLEARMENU	293
26297	CK1NOLASTWD	197	2F2DC	ClearSelection	306
26D006	^CK1TONOext	421	26021	CLEARVDISP	277
184006	^CK1Z	198, 328	2A085	Clipboard!	307
262BA	CK2	196	2A0A5	Clipboard0	307
26305	CK2&Dispatch	197	2A0B5	Clipboard?	307
190006	^CK2FPOLY	326	2A095	Clipboard@	307
2629C	CK2NOLASTWD	197	39144	xCLKADJ	455
185006	^CK2Z	198, 328	2F153	CLKADJ*	
262BF	CK3	196	2EED7	CLKTICKS	173
2630A	CK3&Dispatch	197	39839	xCLLCD	455
262A1	CK3NOLASTWD	197	3EC95	xCLOSEIO	455
186006	^CK3Z	198, 328	2EEC9	CLOSEUART	180
262C4	CK4	196	2F15E	Clr16	277
2630F	CK4&Dispatch	197	2EED4	Clr8	277
262A6	CK4NOLASTWD	197	2EED5	Clr8-15	277
262C9	CK5	196	2571E	ClrAlgEntry	278
26314	CK5&Dispatch	197	26035	ClrAlphaAnn	278
262AB	CK5NOLASTWD	197	2569A	ClrAppMode	224
1CD006	^ckaddt*	361	25676	ClrAppSuspOK	224
1CE006	^ckaddt+	360	2649F	ClrBusyAnn	278
1CF006	^ckaddt-	361	09D007	^CLRCOMPLEX	406
181006	^CKALG	201	2EE74	ClrDA1Bad	276
15A006	^CKCARCOMP	73	2EE7D	ClrDA1IsStat	273
2F321	CkChr00	55	2EE8D	ClrDA10K	275
2BF1C	CkEQUtil		2EE75	ClrDA2aBad	276
18F006	^CKFPOLYext	326	2EE8E	ClrDA2aOK	275
2F324	CKGROBFITS	90	2EEB3	ClrDA2bBad	276
158006	^CKINNERCOMP	72	2EE80	ClrDA2bIsEdL	276
19C006	^CKINT>0	334	2EE81	ClrDA2bNoCh	276
521006	^CKLN	348	2EE8F	ClrDA2bOK	275
177006	^CKMATRIXELEM	326	2EEA7	ClrDA2bTemp	275
262CE	CKN	196	2EE90	ClrDA2OK	275
25F25	CKNNOLASTWD	197	2EEB5	ClrDA3Bad	276
172006	^CKNUMARRY	65	2EE6E	ClrDA3OK	275
2BCA2	cknumdsptch1	87	2EE6D	ClrDAsOK	275
2EF06	CKPICT	316	2EF68	ClrDouseAlm	
36B7B	CKREAL	198	0A1007	^CLREXACT	406
25E9F	CKREF	171	319C1	CLRFRC	
16F006	^CKSAMESIZE	339	2603A	ClrLeftAnn	277
2A7A7	CkSecoType		257BE	ClrNewEditL	
3EDAC	xCKSM	455	2EEAF	ClrNoRollDA2	276
3D2B4	CKSYMBTYPE	199	2561C	ClrNUsrKeyOK	208
193006	^CKSYMREALCMP	201	0AC007	^CLRPLUSAT0	407
261C0	CLCD10	277	2603F	ClrRightAnn	277
19E006	^CLEANIDLAM	326	2F325	ClrServMode	
			1		

26044 ClrSysFlag 175 36965 COLATTE 139 26049 ClrUserFlag 175 359C8 COLANTCase 140 3DDBE XCLSIGMA 455 34AF4 COLARFITE 138 3E91A XCLUSR 455 363FB COLARKIP 131 2EFFB CMD_BAK 305 2EAB3 COLACKIP 131 2EFFB CMD_BAK 305 2EAB3 COLACKIP 131 2EFFB CMD_BAK 305 2EAB3 COLACKIP 131 2EFFB CMD_DAK 306 270006 COLC2 2F3FA CMD_COPY.SBR 307 271006 COLC2 2EFFB CMD_DEB_LINE 305 3E5A0 XCOLCT 455 2EFFD CMD_DEB_LINE 305 3E5A0 XCOLCT 455 2EFF0 CMD_DED 302 0300DE XCOLLECT 455 2EFF1 CMD_DED 302 0300DE XCOLLECT 455 2EFF1 CMD_DROP 302 0BE006 CombineFac 376 2EFF1 CMD_DROP 302 0BE006 CombineFac 376 2EFF1 CMD_DROP 305 0BE006 CombineFac 376 2EFF1 CMD_PAGED 305 0BE006 CombineFac 376 2EFF7 CMD_PAGEL 305 0C1006 COMBINEX 376 2EFF7 CMD_PAGEL 305 0C1006 COMBINEX 376 2EFF7 CMD_PAGEL 305 0C1006 COMBINEX 312 2EFF8 CMD_PAGEL 305 0C1006 COMPLEX 446 2EFF7 CMD_UP 305 0C1006 COMPLEX 312 2EFF8 CMD_PLUS2 301 58B006 COMPLEX 406 2EFF8 CMD_PLUS3 302 099007 COMPLEX 406 2EFF8 CMD_UP 305 275006 COMPLEX 381 3F11C XCMDAPPLY COMPLEX 301 3BP7 XCON 456 3311B cmp 10 3C967 XCONFLEX 381 3F11C XCMDAPPLY 2F0006 COMPLEX 381 3F11C XCMDAPPLY 2F0006 COMPLEX 381 3F11C XCMDAPPLY 3050 XCONFT 456 3311B cmp 10 3C967 XCONFLEX 381 3F11C XCMDAPPLY 3050 XCONFT 456 3311B cmp 10 3C967 XCONFLEX 381 3F11C XCMDAPPLY 3050 XCONFT 456 3311B cmp 10 3C967 XCONFLEX 381 3F11C XCMDAPPLY 3050 XCONFT 456 3311B CMD 1000 XCONFT 456 3311B XCMPL 3000 XCONFT 456 3311B XC	Addr.	Name	Page	Addr.	Name	Page
3DD8E   XCLSIGMA   455   34AF4   COLARPITE   138   3E91A   XCLUSR   455   363FB   COLASKIP   131   2EF7B   CMD_BAK   305   2BAB3   COLAThexFCN   2EF8A   CMD_COPY   306   270006   COLC1   2F2FA   CMD_COPY.SBR   307   271006   COLC2   2F2FA   CMD_COPY.SBR   307   271006   COLC2   2EF8B   CMD_CUT   306   26E006   COLCExt   355   2EF7D   CMD_DEB_LINE   305   3E5A0   XCOLCT   455   2EF82   CMD_DUN   305   3B423   XCOME   456   2EF81   CMD_DOWN   305   3B423   XCOME   456   2EF81   CMD_DOWN   305   3B423   XCOME   456   2EF76   CMD_DOWN   305   30E006   Combinefac   376   2EF77   CMD_PAGED   305   0C1006   Combinet   376   2EF77   CMD_PAGED   305   0BF006   CombProd   376   2EF77   CMD_PAGED   305   0BF006   CombProd   376   2EF77   CMD_PAGED   305   2EF9A   CommandLineHeight   312   2EF78   CMD_PAGEL   305   2EF9A   CommandLineHeight   312   2EF78   CMD_PAGEL   305   2EF9A   CommandLineHeight   312   2EF79   CMD_PAGEU   305   262FB   COMPURAL   128   2EF74   CMD_PLUS   301   09B007   COMPLEX?   406   2EF94   CMD_PLUS   301   SBB006   COMPLEX   406   2EF94   CMD_PLUS   301   SBB006   COMPLEX   406   2EF94   CMD_STO_PEBUT   306   095007   COMPLEXOFF   406   2EF94   CMD_STO_PEBUT   306   095007   COMPLEXOFF   406   2EF74   CMD_STO_FIN   306   095007   COMPLEXOFF   406   2EF74   CMD_STO_FIN   306   095007   COMPLEXOFF   406   2EF75   CMD_UP   305   275006   COMPLINEX   381   3711C   XCMDAPPLY   2FD006   COMPLEXOFF   406   2EF74   CMD_STO_FIN   306   095007   COMPLEXOFF   406   3311B   cmp   10   3C967   XCON   456   3311B   CMD   2 CORCE   22   22006   CONTAINS_LN?   424   250AB   XCON   456   25244   CORCEE2   22   26006   CONTAINS_LN?   424   250AB   XCON   456	26044	ClrSysFlag	175	36865	COLAITE	139
3E91A   XCLUSR   305   2BAB3   COLASKIP   131	26049	ClrUserFlag	175	359C8	COLANOTcase	140
2EF7B	3DD8E	xCLSIGMA	455	34AF4	COLARPITE	138
2EF8A	3E91A	xCLUSR	455	363FB	COLASKIP	131
2F2FA CMD_COPY.SBR 307 271006	2EF7B	CMD_BAK	305	2BAB3	COLAthexFCN	
2EF88	2EF8A	CMD_COPY	306	270006	^COLC1	
2EF7D         CMD_DEB_LINE         305         3E5A0         XCOLCT         455           2EF82         CMD_DEL         302         3020DE         XCOLLECT         455           2EF80         CMD_DOWN         305         3B423         XCOMB         456           2EF81         CMD_DOROP         302         0BE006         ^CombineFac         376           2EF7E         CMD_END_LINE         305         0C0006         ^CcmbInt         376           2EF7A         CMD_PAGED         305         0C1006         ^CcmbProd         376           2EF7A         CMD_PAGEL         305         0BF006         ^CcmbProd         376           2EF77         CMD_PAGEL         305         2EF9A         CommandLineHeight         312           2EF79         CMD_PAGEU         305         262PB         COMPEVAL         128           2EF79         CMD_PAGEU         301         09B007         ^CCMPLEX         406           2EF14         CMD_PLUS2         301         09B007         ^CCMPLEXRR         404           2F195         CMD_PLUS2         301         09B007         ^CCMPLEXMODE         406           2EF86         CMD_STO_DEBUT         306         0	2F2FA	CMD_COPY.SBR	307	271006	^COLC2	
2EF82         CMD_DEL         302         0300DE         xCOLLECT         455           2EF80         CMD_DOWN         305         3B423         xCOMB         456           2EF81         CMD_DROP         302         0BE006         ^CcombineFac         376           2EF7E         CMD_END_LINE         305         0C1006         ^Ccombnext         376           2EF7C         CMD_NXT         305         0BF006         ^Ccombnext         376           2EF7A         CMD_PAGED         305         0BF006         ^Ccombrext         376           2EF77         CMD_PAGEL         305         2EF9A         CommandLineHeight         312           2EF78         CMD_PAGER         305         0DD100         xCOMP→         478           2EF78         CMD_PAGEU         305         262FB         COMMPEVAL         128           2EF79         CMD_PAGEU         305         262FB         COMPLEXCH         128           2EF79         CMD_PLUS         301         988007         ^COMPLEXCR         406           2F194         CMD_PLUS3         302         099007         ^COMPLEXCR         406           2F195         CMD_PLUS3         302         099007<	2EF88	CMD_CUT	306	26E006	^COLCext	355
2EF80	2EF7D	CMD_DEB_LINE	305	3E5A0	XCOLCT	455
2EF81         CMD_DROP         302         0BE006         ^CombineFac         376           2EF7C         CMD_END_LINE         305         0C0006         ^CombInit         376           2EF7C         CMD_PAGED         305         0C1006         ^CombPexed         376           2EF7A         CMD_PAGEL         305         0EF006         ^CombProd         376           2EF77         CMD_PAGEL         305         2EF9A         CommandLineHeight         312           2EF79         CMD_PAGER         305         0D1100         xCOMP→         478           2EF79         CMD_PAGEU         305         262FB         COMPEVAL         128           2EF79         CMD_PLUS         301         098007         ^COMPLEXERR         406           2F194         CMD_PLUS2         301         588006         ^COMPLEXORF         406           2F194         CMD_PLUS3         302         099007         ^COMPLEXORF         406           2F195         CMD_PLUS3         302         099007         ^COMPLEXORF         406           2EF84         CMD_STO_FIN         306         095007         ^COMPLEXORF         406           2EF84         CMD_STO_FIN         306	2EF82	CMD_DEL	302	0300DE	xCOLLECT	455
ZEF7E         CMD_END_LINE         305         OC0006         ^CombInit         376           ZEF7C         CMD_NXT         305         OC1006         ^CombNext         376           ZEF7A         CMD_PAGED         305         OBF006         ^CombProd         376           ZEF77         CMD_PAGEL         305         ZEF9A         CommandLineHeight         312           ZEF78         CMD_PAGER         305         000100         xCOMP→         478           ZEF79         CMD_PAGEU         305         262FB         COMPEVAL         128           ZEF74         CMD_PLUS         301         098007         ^COMPLEXERR         404           2F194         CMD_PLUS2         301         588006         ^COMPLEXERR         404           2F195         CMD_PLUS3         302         099007         ^COMPLEXOFF         406           2EF84         CMD_STO_DEBUT         306         095007         ^COMPLEXOFF         406           2EF75         CMD_UP         305         275006         ^COMPLEXON         406           2EF76         CMD_UP         305         275006         ^COMPLEXON         406           2EF77         CMD_UP         305	2EF80	CMD_DOWN	305	3B423	xCOMB	456
2EF7C         CMD_NXT         305         OC1006         ^CombNext         376           2EF7A         CMD_PAGED         305         OBF006         ^CombProd         376           2EF77         CMD_PAGEL         305         2EF9A         CommandLineHeight         312           2EF78         CMD_PAGER         305         00D100         xCOMP→         478           2EF79         CMD_PAGEU         305         262FB         COMPEVAL         128           2EF74         CMD_PLUS         301         098007         ^COMPLEXR         406           2F194         CMD_PLUS2         301         58B006         ^COMPLEXRR         404           2F195         CMD_PLUS3         302         099007         ^COMPLEXRR         406           2EF83         CMD_STO_DEBUT         306         095007         ^COMPLEXON         406           2EF76         CMD_UP         305         275006         ^COMPLEXON         406           2EF77         CMD_OPLY         306         095007         ^COMPLEXON         406           2EF77         CMD_OPLY         306         095007         ^COMPLEXON         406           2EF84         CMD_STO_FIN         306         0	2EF81	CMD_DROP	302	0BE006	^CombineFac	376
2EF7A         CMD_PAGED         305         DBF006         ^CombProd         376           2EF77         CMD_PAGEL         305         2EF9A         CommandLineHeight         312           2EF78         CMD_PAGER         305         00D100         xCOMP→         478           2EF79         CMD_PAGEU         305         262FB         COMPEVAL         128           2EF74         CMD_PLUS         301         09B007         ^COMPLEXER         406           2F194         CMD_PLUS2         301         58B006         ^COMPLEXERR         404           2F195         CMD_STO_DEBUT         306         096007         ^COMPLEXON         406           2EF83         CMD_STO_DEBUT         306         095007         ^COMPLEXON         406           2EF84         CMD_STO_FIN         306         095007         ^COMPLEXON         406           2EF76         CMD_UP         305         275006         ^COMPLEXON         406           2EF77         CMD_UP         305         275006         ^COMPLEXON         406           2EF784         CMD_STO_DEBUT         306         095007         ^COMPLEXON         436           2E7816         CMD_UP         305	2EF7E	CMD_END_LINE	305	0C0006	^CombInit	376
2EF77         CMD_PAGEL         305         2EF9A         CommandLineHeight         312           2EF78         CMD_PAGER         305         00D100         xCOMP→         478           2EF79         CMD_PAGEU         305         262FB         COMPEVAL         128           2EF74         CMD_PLUS         301         098007         ^COMPLEXER         406           2F194         CMD_PLUS3         302         099007         ^COMPLEXOFF         406           2EF83         CMD_STO_DEBUT         306         096007         ^COMPLEXOFF         406           2EF84         CMD_STO_FIN         306         095007         ^COMPLEXOFF         406           2EF7F         CMD_UP         305         275006         ^COMPLEXOFF         406           2EF7F         CMD_UP         305         275006         ^COMPLEXOFF         406           2EF7F         CMD_UP         306         095007         ^COMPLEXOFF         406           2EF84         CMD_STO_TINB         306         096007         ^COMPLEXOFF         406           2EF84         CMD_STO_TINB         306         095007         ^COMPLEXOF         381           3F11         XCMDAPPLY         XCOMDAPP<	2EF7C	CMD_NXT	305	0C1006	^CombNext	376
2EF78         CMD_PAGER         305         00D100         xCOMP→         478           2EF79         CMD_PAGEU         305         262FB         COMPEVAL         128           2EF74         CMD_PLUS         301         09B007         ^COMPLEXP         406           2F194         CMD_PLUS2         301         58B006         ^COMPLEXERR         404           2F195         CMD_PLUS3         302         099007         ^COMPLEXOFF         406           2EF83         CMD_STO_DEBUT         306         095007         ^COMPLEXOFF         406           2EF84         CMD_STO_FIN         306         095007         ^COMPLEXON         406           2EF7F         CMD_UP         305         275006         ^COMPLEXON         456           2B6006         ^CMDETA         426         026000	2EF7A	CMD_PAGED	305	0BF006	^CombProd	376
2EF79         CMD_PAGEU         305         262FB         COMPEVAL         128           2EF74         CMD_PLUS         301         09B007         ^COMPLEX?         406           2F194         CMD_PLUS2         301         58B006         ^COMPLEXERR         406           2F195         CMD_PLUS3         302         099007         ^COMPLEXOFF         406           2EF83         CMD_STO_FIN         306         095007         ^COMPLEXOFF         406           2EF84         CMD_STO_FIN         306         095007         ^COMPLEXON         406           2EF7F         CMD_UP         305         275006         ^COMPLISTON         456           2B6006         CMDEXT         426         226008	2EF77	CMD_PAGEL	305	2EF9A	CommandLineHeight	312
2EF74         CMD_PLUS         301         09B007         ^COMPLEX?         406           2F194         CMD_PLUS2         301         58B006         ^COMPLEXERR         404           2F195         CMD_PLUS3         302         099007         ^COMPLEXMODE         406           2EF83         CMD_STO_DEBUT         306         096007         ^COMPLEXMODE         406           2EF84         CMD_STO_FIN         306         095007         ^COMPLEXON         406           2EF7F         CMD_UP         305         275006         ^COMPLEXON         406           2EF7F         CMD_STO_DEBUT         305         275006         ^COMPLEXON         406           2EF326         CMDSTO         314         38F7         xCON         456           2B0006         ^CONPLEXION         345         3980	2EF78	CMD_PAGER	305	00D100	$xCOMP \rightarrow$	478
2F194         CMD_PLUS2         301         58B006         ^COMPLEXERR         404           2F195         CMD_PLUS3         302         099007         ^COMPLEXMODE         406           2EF83         CMD_STO_DEBUT         306         096007         ^COMPLEXON         406           2EF84         CMD_STO_FIN         306         095007         ^COMPLEXON         406           2EF7F         CMD_UP         305         275006         ^COMPLISTEXT         381           3F11C         XCMDAPPLY         2FD006         ^COMPRIMEXT         73           2F326         CMDSTO         314         3BF77         XCON         456           2B6006         ^CMDDEXT         424         0260AB         XCOND         456           3311B         cmp         10         3C967         XCONIC         456           081314         ^XCMPLX         455         39A6C         XCONJ         456           518006         ^CNORMEXT         349         0180AB         XCONLIB         456           518006         ^CNORMEXT         333         3396D         Connecting         20           3B193         XCMM         455         0190AB         XCONSTANTE <t< td=""><td>2EF79</td><td>CMD_PAGEU</td><td>305</td><td>262FB</td><td>COMPEVAL</td><td>128</td></t<>	2EF79	CMD_PAGEU	305	262FB	COMPEVAL	128
2F195         CMD_PLUS3         302         099007         ^COMPLEXMODE         406           2EF83         CMD_STO_DEBUT         306         096007         ^COMPLEXOFF         406           2EF84         CMD_STO_FIN         306         095007         ^COMPLEXON         406           2EF7F         CMD_UP         305         275006         ^COMPLISTEXT         381           3F11C         XCMDAPPLY         2FD006         ^COMPRIMEXT         73           2F326         CMDSTO         314         3BF77         XCON         456           2B6006         ^CMDEXT         424         0260AB         XCOND         456           3311B         cmp         10         3C967         XCONIC         456           081314         ^XCMPLX         455         39A6C         XCONJ         456           526006         ^CMPLXLN         349         0180AB         XCONLIB         456           518006         ^CNORMEXT         333         3396D         Connecting         20           3B193         XCRM         455         0190AB         XCONST         456           25EA9         COdeP1>%rc.p         205         39B1E         XCONSTANTS         45	2EF74	CMD_PLUS	301	09В007	^COMPLEX?	406
2EF83         CMD_STO_DEBUT         306         096007         ^COMPLEXOFF         406           2EF84         CMD_STO_FIN         306         095007         ^COMPLEXON         406           2EF7F         CMD_UP         305         275006         ^COMPLISTEXT         381           3F11C         XCMDAPPLY         ZF0006         ^COMPLISTEXT         73           2F326         CMDSTO         314         3BF77         XCON         456           2B6006         ^CMODEXT         424         0260AB         XCOND         456           3311B         cmp         10         3C967         XCONIC         456           081314         ^XCMPLX         455         39A6C         XCONJ         456           526006         ^CMPLXLN         349         0180AB         XCONLIB         456           518006         ^CNORMEXT         333         3396D         Connecting         20           3B193         XCNM         455         0190AB         XCONST         456           25EA9         COGEPL>%rc.p         205         39B1E         XCONSTANTS         456           25EA0         COERCE\$22         47         3989C         XCONT         456	2F194	CMD_PLUS2	301	58B006	^COMPLEXERR	404
2EF84         CMD_STO_FIN         306         095007         ^COMPLEXON         406           2EF7F         CMD_UP         305         275006         ^COMPLISTEXT         381           3F11C         xCMDAPPLY         2FD006         ^COMPLISTEXT         73           2F326         CMDSTO         314         3BF77         xCON         456           2B6006         ^CMODEXT         424         0260AB         xCOND         456           3311B         cmp         10         3C967         xCONIC         456           081314         ^xCMPLX         455         39A6C         xCONJ         456           526006         ^CMPLXLN         349         0180AB         xCONLIB         456           518006         ^CNORMEXT         333         3396D         Connecting         20           3B193         xCNRM         455         0190AB         xCONST         456           25EA9         CodePl>%rc.p         205         39B1E         xCONSTANTS         458           262F1         COERCE\$22         47         3989C         xCONT         456           2F244         COERCE\$22         47         3989C         xCONT         456	2F195	CMD_PLUS3	302	099007	^COMPLEXMODE	406
2EF7F         CMD_UP         305         275006         ^COMPLISTEXT         381           3F11C         xCMDAPPLY         2FD006         ^COMPRIMEXT         73           2F326         CMDSTO         314         3BF77         xCON         456           2B6006         ^CMODEXT         424         0260AB         xCOND         456           3311B         cmp         10         3C967         xCONIC         456           081314         ^xCMPLX         455         39A6C         xCONJ         456           526006         ^CMPLXLN         349         0180AB         xCONLIB         456           518006         ^CNORMEXT         333         3396D         Connecting         20           3B193         xCNRM         455         0190AB         xCONSTANTE         456           25EA9         CodePl>%rc.p         205         39B1E         xCONSTANTE         458           262F1         COERCE         22         02A0DE         xCONT         456           2F244         COERCE\$22         47         3989C         xCONT         456           2F244         COERCEEZ         22         2CB006         ^CONTAINS_LN?         424	2EF83	CMD_STO_DEBUT	306	096007	^COMPLEXOFF	406
3F11C       xCMDAPPLY       2FD006       ^COMPRIMEXT       73         2F326       CMDSTO       314       3BF77       xCON       456         2B6006       ^CMODEXT       424       0260AB       xCOND       456         3311B       cmp       10       3C967       xCONIC       456         081314       ^xCMPLX       455       39A6C       xCONJ       456         526006       ^CMPLXLN       349       0180AB       xCONLIB       456         518006       ^CNORMEXT       333       3396D       Connecting       20         3B193       xCNRM       455       0190AB       xCONST       456         25EA9       CodePl>*rc.p       205       39B1E       xCONSTANTE       458         262F1       COERCE       22       02A0DE       xCONTAINTS       456         25EA0       COERCE\$22       47       3989C       xCONT       456         2F244       COERCE&CKSGN       22       0FF006       ^CONTAINS_LN?       424         0F0006       ^COERCE2Z       22       8071B       CONTEXT       426         35D08       COERCEBAP       22       08D08       CONTEXT@       169 </td <td>2EF84</td> <td>CMD_STO_FIN</td> <td>306</td> <td>095007</td> <td>^COMPLEXON</td> <td>406</td>	2EF84	CMD_STO_FIN	306	095007	^COMPLEXON	406
2F326       CMDSTO       314       3BF77       xCON       456         2B6006       ^CMODext       424       0260AB       xCOND       456         3311B       cmp       10       3C967       xCONIC       456         081314       ^xCMPLX       455       39A6C       xCONJ       456         526006       ^CMPLXLN       349       0180AB       xCONLIB       456         518006       ^CNORMext       333       3396D       Connecting       20         3B193       xCNRM       455       0190AB       xCONST       456         25EA9       CodePl>%rc.p       205       39B1E       xCONSTANTE       458         262F1       COERCE       22       02A0DE       xCONTANTS       456         25EA0       COERCE\$22       47       3989C       xCONT       456         2F244       COERCE&CKSGN       22       0FF006       ^CONTAINS_LN?       424         0F0006       ^COERCE2Z       22       20B008       CONTEXT       426         2602B       COERCEDUP       22       08D08       CONTEXT@       169         2502B       COERCEFLAG       135       08D5A       CONTEXT@ <t< td=""><td>2EF7F</td><td>CMD_UP</td><td>305</td><td>275006</td><td>^COMPLISText</td><td>381</td></t<>	2EF7F	CMD_UP	305	275006	^COMPLISText	381
2B6006       ^CMODEXT       424       0260AB       xCOND       456         3311B       cmp       10       3C967       xCONIC       456         081314       ^xCMPLX       455       39A6C       xCONJ       456         526006       ^CMPLXLN       349       0180AB       xCONLIB       456         518006       ^CNORMEXT       333       3396D       Connecting       20         3B193       xCNRM       455       0190AB       xCONST       456         25EA9       CodePl>*rc.p       205       39B1E       xCONSTANTE       458         262F1       COERCE       22       02A0DE       xCONSTANTS       456         25EA0       COERCE\$22       47       3989C       xCONT       456         2F244       COERCE\$22       22       0FF006       ^CONTAINS_LN?       424         0F0006       ^CCERCE2       22       2CB006       ^CONTAINS_LN?       424         0F0006       ^CCERCE2Z       22       8071B       CONTEXT?       169         2602B       COERCEFLAG       135       08D5A       CONTEXT@       169         35EB6       COERCESWAP       22       204006       ^CONVBACK2	3F11C	xCMDAPPLY		2FD006	^COMPRIMext	73
3311B       cmp       10       3C967       xCONIC       456         081314       ~xCMPLX       455       39A6C       xCONJ       456         526006       ^CMPLXLN       349       0180AB       xCONLIB       456         518006       ^CNORMext       333       3396D       Connecting       20         3B193       xCNRM       455       0190AB       xCONST       456         25EA9       CodePl>%rc.p       205       39B1E       xCONSTANTE       458         262F1       COERCE       22       02A0DE       xCONSTANTS       58         25EA0       COERCE\$22       47       3989C       xCONT       456         2F244       COERCE&CKSGN       22       0FF006       ^CONTains?       137         3F481       COERCE2       22       2CB006       ^CONTAINS_LN?       424         0F0006       ^COERCEZZ       22       8071B       CONTEXT       169         2602B       COERCEDUP       22       08D08       CONTEXT@       169         35EB6       COERCESWAP       22       204006       ^CONVBACKINT       328         03F0AB       xCOL+       455       38F41       xCONVERT	2F326	CMDSTO	314	3BF77	xCON	456
081314       "xCMPLX       455       39A6C       xCONJ       456         526006       ^CMPLXLN       349       0180AB       xCONLIB       456         518006       ^CNORMext       333       3396D       Connecting       20         3B193       xCNRM       455       0190AB       xCONST       456         25EA9       CodePl>*rc.p       205       39B1E       xCONSTANTE       458         262F1       COERCE       22       02A0DE       xCONTANTS       456         25EA0       COERCE\$22       47       3989C       xCONT       456         2F244       COERCE&CKSGN       22       0FF006       ^CONTAINS_LN?       424         0F0006       ^COERCE2       22       2CB006       ^CONTAINS_LN?       424         0F0006       ^COERCEZZ       22       8071B       CONTEXT       459         35D08       COERCEDUP       22       08D08       CONTEXT@       169         35EB6       COERCESWAP       22       204006       ^CONVBACK2INT       328         03F0AB       xCOL+       455       38F41       xCONVERT       456         0390AB       xCOL→       455       2F327       c	2B6006	^CMODext	424	0260AB	xCOND	456
526006       ^CMPLXLN       349       0180AB       xCONLIB       456         518006       ^CNORMext       333       3396D       Connecting       20         3B193       xCNRM       455       0190AB       xCONST       456         25EA9       CodePl>%rc.p       205       39B1E       xCONSTANTE       458         262F1       COERCE       22       02A0DE       xCONSTANTS       456         25EA0       COERCE\$22       47       3989C       xCONT       456         2F244       COERCE&CKSGN       22       0FF006       ^CONTAINS_LN?       424         0F0006       ^COERCE2Z       22       2CB006       ^CONTEXT       424         0F0006       ^COERCEDUP       22       08D08       CONTEXT       169         2602B       COERCEFLAG       135       08D5A       CONTEXT@       169         35EB6       COERCESWAP       22       204006       ^CONVBACK2INT       328         03F0AB       xCOL+       455       38F41       xCONVERT       456         0390AB       xCOL→       455       2F327       convertbase         06FD1       COLA       131       2C393       COPYVAR     <	3311B	cmp	10	3C967	xCONIC	456
518006       ^CNORMext       333       3396D       Connecting       20         3B193       xCNRM       455       0190AB       xCONST       456         25EA9       CodePl>%rc.p       205       39B1E       xCONSTANTE       458         262F1       COERCE       22       02A0DE       xCONSTANTS       456         25EA0       COERCE\$22       47       3989C       xCONT       456         2F244       COERCE&CKSGN       22       0FF006       ^CONTAINS_LN?       424         0F0006       ^COERCE2       22       2CB006       ^CONTAINS_LN?       424         0F0006       ^COERCE2Z       22       8071B       CONTEXT       169         2602B       COERCEDUP       22       08D08       CONTEXT@       169         35EB6       COERCESWAP       22       204006       ^CONVBACK2INT       328         03F0AB       xCOL+       455       38F41       xCONVERT       456         0390AB       xCOL→       455       2F327       convertbase         06FD1       COLA       131       2C393       COPYVAR         34AD3       COLA_EVAL       131       3DE24       xCOR       456 <td>081314</td> <td>~xCMPLX</td> <td>455</td> <td>39A6C</td> <td>xCONJ</td> <td>456</td>	081314	~xCMPLX	455	39A6C	xCONJ	456
3B193       xCNRM       455       0190AB       xCONST       456         25EA9       CodePl>%rc.p       205       39B1E       xCONSTANTE       458         262F1       COERCE       22       02A0DE       xCONSTANTS         25EA0       COERCE\$22       47       3989C       xCONT       456         2F244       COERCE&CKSGN       22       0FF006       ^CONTains?       137         3F481       COERCE2       22       2CB006       ^CONTAINS_LN?       424         0F0006       ^COERCE2Z       22       8071B       CONTEXT       169         35D08       COERCEDUP       22       08D08       CONTEXT!       169         2602B       COERCEFLAG       135       08D5A       CONTEXT@       169         35EB6       COERCESWAP       22       204006       ^CONVBACK2INT       328         03F0AB       xCOL+       455       38F41       xCONVERT       456         0390AB       xCOL→       455       2F327       convertbase         06FD1       COLA       131       2C393       COPYVAR         34AD3       COLA_EVAL       131       3DE24       xCOR       456         359A	526006	^CMPLXLN	349	0180AB	xCONLIB	456
25EA9       CodePl>%rc.p       205       39B1E       xCONSTANTE       458         262F1       COERCE       22       02A0DE       xCONSTANTS         25EA0       COERCE\$22       47       3989C       xCONT       456         2F244       COERCE&CKSGN       22       0FF006       ^Contains?       137         3F481       COERCE2       22       2CB006       ^CONTAINS_LN?       424         0F0006       ^COERCE2Z       22       8071B       CONTEXT       169         35D08       COERCEDUP       22       08D08       CONTEXT!       169         2602B       COERCEFLAG       135       08D5A       CONTEXT@       169         35EB6       COERCESWAP       22       204006       ^CONVBACK2INT       328         03F0AB       xCOL+       455       205006       ^CONVBACKINT       328         03F0AB       xCOL-       455       38F41       xCONVERT       456         03F0AB       xCOL-       455       2F327       convertbase         06FD1       COLA       131       2C393       COPYVAR         34AD3       COLA_EVAL       131       3DE24       xCOR       456         <	518006	^CNORMext	333	3396D	Connecting	20
262F1       COERCE       22       02A0DE       xCONSTANTS         25EA0       COERCE\$22       47       3989C       xCONT       456         2F244       COERCE&CKSGN       22       0FF006       ^Contains?       137         3F481       COERCE2       22       2CB006       ^CONTAINS_LN?       424         0F0006       ^COERCEZZ       22       8071B       CONTEXT       169         35D08       COERCEDUP       22       08D08       CONTEXT!       169         2602B       COERCEFLAG       135       08D5A       CONTEXT@       169         35EB6       COERCESWAP       22       204006       ^CONVBACK2INT       328         03F0AB       xCOL+       455       205006       ^CONVBACKINT       328         03E0AB       xCOL-       455       38F41       xCONVERT       456         03F0AB       xCOL-       455       2F327       convertbase         06FD1       COLA       131       2C393       COPYVAR         34AD3       COLA_EVAL       131       3DE24       xCORR       456         359AD       COLAcase       140       3A5D0       xCOS       456	3B193	xCNRM	455	0190AB	xCONST	456
25EA0         COERCE\$22         47         3989C         xCONT         456           2F244         COERCE&CKSGN         22         0FF006         ^Contains?         137           3F481         COERCE2         22         2CB006         ^CONTAINS_LN?         424           0F0006         ^COERCE2Z         22         8071B         CONTEXT         169           35D08         COERCEDUP         22         08D08         CONTEXT!         169           2602B         COERCEFLAG         135         08D5A         CONTEXT@         169           35EB6         COERCESWAP         22         204006         ^CONVBACK2INT         328           03F0AB         xCOL+         455         205006         ^CONVBACKINT         328           03F0AB         xCOL-         455         38F41         xCONVERT         456           0390AB         xCOL-         455         2F327         convertbase           06FD1         COLA         131         2C393         COPYVAR           34AD3         COLA_EVAL         131         3DE24         xCORR         456           359AD         COLAcase         140         3A5D0         xCOS         456	25EA9	CodePl>%rc.p	205	39B1E	xCONSTANTe	458
2F244       COERCE&CKSGN       22       OFF006       ^Contains?       137         3F481       COERCE2       22       2CB006       ^CONTAINS_LN?       424         0F0006       ^COERCE2Z       22       8071B       CONTEXT       169         35D08       COERCEDUP       22       08D08       CONTEXT!       169         2602B       COERCEFLAG       135       08D5A       CONTEXT@       169         35EB6       COERCESWAP       22       204006       ^CONVBACK2INT       328         03F0AB       xCOL+       455       205006       ^CONVBACKINT       328         03E0AB       xCOL-       455       38F41       xCONVERT       456         0390AB       xCOL→       455       2F327       convertbase         06FD1       COLA       131       2C393       COPYVAR         34AD3       COLA_EVAL       131       3DE24       xCORR       456         359AD       COLAcase       140       3A5D0       xCOS       456	262F1	COERCE	22	02A0DE	xCONSTANTS	
3F481       COERCE2       22       2CB006       ^CONTAINS_LN?       424         0F0006       ^COERCE2Z       22       8071B       CONTEXT         35D08       COERCEDUP       22       08D08       CONTEXT!       169         2602B       COERCEFLAG       135       08D5A       CONTEXT@       169         35EB6       COERCESWAP       22       204006       ^CONVBACK2INT       328         03F0AB       xCOL+       455       205006       ^CONVBACKINT       328         03E0AB       xCOL-       455       38F41       xCONVERT       456         0390AB       xCOL→       455       2F327       convertbase         06FD1       COLA       131       2C393       COPYVAR         34AD3       COLA_EVAL       131       3DE24       xCORR       456         359AD       COLAcase       140       3A5D0       xCOS       456	25EA0	COERCE\$22	47	3989C	xCONT	456
0F0006         ^COERCE2Z         22         8071B         CONTEXT           35D08         COERCEDUP         22         08D08         CONTEXT!         169           2602B         COERCEFLAG         135         08D5A         CONTEXT@         169           35EB6         COERCESWAP         22         204006         ^CONVBACK2INT         328           03F0AB         xCOL+         455         205006         ^CONVBACKINT         328           03E0AB         xCOL-         455         38F41         xCONVERT         456           0390AB         xCOL→         455         2F327         convertbase           06FD1         COLA         131         2C393         COPYVAR           34AD3         COLA_EVAL         131         3DE24         xCORR         456           359AD         COLAcase         140         3A5D0         xCOS         456	2F244	COERCE&CKSGN	22	0FF006	^Contains?	137
35D08       COERCEDUP       22       08D08       CONTEXT!       169         2602B       COERCEFLAG       135       08D5A       CONTEXT@       169         35EB6       COERCESWAP       22       204006       ^CONVBACK2INT       328         03F0AB       xCOL+       455       205006       ^CONVBACKINT       328         03E0AB       xCOL-       455       38F41       xCONVERT       456         0390AB       xCOL→       455       2F327       convertbase         06FD1       COLA       131       2C393       COPYVAR         34AD3       COLA_EVAL       131       3DE24       xCORR       456         359AD       COLAcase       140       3A5D0       xCOS       456	3F481	COERCE2	22	2CB006	^CONTAINS_LN?	424
2602B       COERCEFLAG       135       08D5A       CONTEXT@       169         35EB6       COERCESWAP       22       204006       ^CONVBACK2INT       328         03F0AB       xCOL+       455       205006       ^CONVBACKINT       328         03E0AB       xCOL-       455       38F41       xCONVERT       456         0390AB       xCOL→       455       2F327       convertbase         06FD1       COLA       131       2C393       COPYVAR         34AD3       COLA_EVAL       131       3DE24       xCORR       456         359AD       COLAcase       140       3A5D0       xCOS       456	0F0006	^COERCE2Z	22	8071B	CONTEXT	
35EB6 COERCESWAP 22 204006 ^CONVBACK2INT 328  03F0AB xCOL+ 455 205006 ^CONVBACKINT 328  03E0AB xCOL− 455 38F41 xCONVERT 456  0390AB xCOL→ 455 2F327 convertbase  06FD1 COLA 131 2C393 COPYVAR  34AD3 COLA_EVAL 131 3DE24 xCORR 456  359AD COLAcase 140 3A5D0 xCOS 456	35D08	COERCEDUP		08D08	CONTEXT!	
03F0AB       xCOL+       455       205006       ^CONVBACKINT       328         03E0AB       xCOL-       455       38F41       xCONVERT       456         0390AB       xCOL→       455       2F327       convertbase         06FD1       COLA       131       2C393       COPYVAR         34AD3       COLA_EVAL       131       3DE24       xCORR       456         359AD       COLAcase       140       3A5D0       xCOS       456	2602B	COERCEFLAG	135	08D5A	CONTEXT@	169
03E0AB       xCOL-       455       38F41       xCONVERT       456         0390AB       xCOL→       455       2F327       convertbase         06FD1       COLA       131       2C393       COPYVAR         34AD3       COLA_EVAL       131       3DE24       xCORR       456         359AD       COLAcase       140       3A5D0       xCOS       456	35EB6	COERCESWAP	22	204006	^CONVBACK2INT	328
0390AB       xCOL→       455       2F327       convertbase         06FD1       COLA       131       2C393       COPYVAR         34AD3       COLA_EVAL       131       3DE24       xCORR       456         359AD       COLAcase       140       3A5D0       xCOS       456	03F0AB	xCOL+	455	205006	^CONVBACKINT	328
06FD1         COLA         131         2C393         COPYVAR           34AD3         COLA_EVAL         131         3DE24         xCORR         456           359AD         COLAcase         140         3A5D0         xCOS         456	03E0AB	xCOL-	455	38F41	xCONVERT	456
34AD3         COLA_EVAL         131         3DE24         xCORR         456           359AD         COLAcase         140         3A5D0         xCOS         456	0390AB	$\texttt{xCOL}{ o}$	455	2F327	convertbase	
359AD COLAcase 140 3A5D0 xCOS 456	06FD1		131	2C393	COPYVAR	
	34AD3	COLA_EVAL	131	3DE24	xCORR	456
35994 COLACOLA 131   42B006 ^cos*tan 366	359AD	COLAcase		3A5D0		456
	35994	COLACOLA	131	42B006	^cos*tan	366

Addr.	Name	Page	Addr.	Name	Page
447006	^cos2exp	367	2657B	CURSOR_OFF0	
446006	^COS2EXPext	352	26594	CURSOR_PART	300
42C006	^COS2ext	351	519006	^CXIRext	
413006	^cos2tan	366	15D006	^CXRIext	333
412006	^COS2TAN	351	0150DE	xCYCLOTOMIC	
409006	^cos2tan/2	365	0120AB	xCYLIN	456
408006	^COS2TAN/2	351	50C006	^CZABS	37
309006	^COSEXPA	355	2C07B	D/D*	
30D006	^COSEXPA*	369	2C086	D/D+	
30E006	^COSEXPA*1	369	2C091	D/D-	
30B006	^COSEXPA+	369	2C09C	D/D/	
30C006	^COSEXPA-	369	2C10B	D/D=	
52C006	^xCOSext	349	2C268	D/D^	
3A6C2	xCOSH	456	2C273	D/D^X	
44B006	^cosh2exp	367	2C27E	D/D^Y	
44A006	^COSH2EXPext	352	2C116	D/DABS	
52F006	^xCOSHext	349	2C13A	D/DACOS	
0CE007	^COSTEST		2C145	D/DACOSH	
3DE3F	xCOV	456	2C150	D/DALOG	
3D128	xCR	456	2C2B5	D/DAPPLY	
01B100	xCRC	479	2C15B	D/DARG	
393CA	xCRDIR	456	2C166	D/DASIN	
08696	CREATE	167	2C171	D/DASINH	
25EA1	CREATEDIR	168	2C17C	D/DATAN	
00113	CRER	100	2C187	D/DATANH	
27195	CRLF\$	43	2C192	D/DCHS	
01A100	xCRLIB	479	2C1B0	D/DCONJ	
3B208	xCROSS	456	2C1CE	D/DCOS	
2EEFA	CROSS_HAIRS		2C1D9	D/DCOSH	
2EEFB	CROSS_OFF		2C289	D/DDER	
27D3F	CROSSGROB	90	2C1E4	D/DEXP	
2F328	CROSSMARKON	00	2C21B	D/DIFTE	
25EA2	CRUNCH	86	2C29F	D/DINTEGRAL	
25EA3	CRUNCHNoBlame	00	2C1EF	D/DINV	
4D5006	^CSQFFext	333	2C1FA	D/DLN	
3EA01	CST	293	2C205	D/DLNP1	
0FF007	^CSTFRACTION?	393	2C210	D/DLOG	
2597B	CtlAlarm!	000	2C226	D/DEGG D/DSIN	
25980	CtlAlarm@		2C220 2C231	D/DSINH	
057314	~xCURL	456	2C23C	D/DSINII D/DSQ	
22E006	^CURL	415	2C23C	D/DSQ D/DSQRT	
2EEFE	CURRENTMARK?	410	2C247	D/DSUM	
860B8	CurROMBank2				
2658A	CURSOR+		2C252 2C25D	D/DTAN	
26585		300	2C25D 2C294	D/DTANH D/DWHERE	
	CURSOR@	300			458
2EEEA	CURSOR_END?		3B06E	xD>R	$\begin{array}{c} 458 \\ 276 \end{array}$
26030	CURSOR_OFF	300	2EEBO	DA1Bad?	
26580	CURSOR_OFF+		2EEAB	DA1IsStatus?	276

Addr.	Name	Page	Addr.	Name	Page
2EE62	DA1OK?	275	3B549	xDEG	456
2BF3A	DA10K?NOTIT	275	288006	^DEG1	382
2EEB1	DA2aBad?	276	285006	^DEG2ext	382
2EE66	DA2aLess1OK?	275	0360DE	xDEGREE	
2BF53	DA2aOK?NOTIT	275	3E0006	^DEGREext	378
2EEB2	DA2bBad?	276	2EF95	DEL_CMD	303
2EE7E	DA2bIsEdL?	276	391D8	xDELALARM	457
2EEB7	DA2bNoCh?	276	3D1C7	xDELAY	457
2BF6C	DA2bOK?NOTIT	275	29D6A	delimcase	
2EEA6	DA2bTemp?		3EF3B	xDELKEYS	457
2EEB4	DA3Bad?	276	0F3007	^DELTAPSOLVE	385
2EE63	DA3OK?	275	0BA006	^DemonicLf	375
2BF85	DA3OK?NOTIT	275	2A5006	^DENOLCMext	358
29EE9	DaDGNTc		3C51F	xDEPND	457
0610AB	xDARCY	456	3DCA7	xDEPTH	457
36E07	dARRYcase	145	0314C	DEPTH	107
39078	XDATE	456	4F2006	^DEPTHext	398
2EED0	DATE	173	4F3006	^DEPTHOBJext	398
39238	xDATE+	456	80EDC	DEPTHSAVE	
2EED2	DATE+DAYS	173	3D258	xDER	
2F329	Date>d\$	173	1B3006	^DERARG	390
2F1AB	Date>hxs13	173	00E314	~xDERIV	457
2DEBB	DAY#		1A5006	^DERIV	388
2DD27	Day>Date		1A1006	^DERIVext	387
0690AB	xdB	456	1A3006	^DERIVIDNT	388
0150DD	xDBUG	456	1A4006	^DERIVIDNT1	388
2F190	DcompWidth@	51	2C0ED	derprod1	
2EED1	DDAYS	173	2C0A7	derquot	
39218	xDDAYS	456	1DD006	^DERVX	412
00C007	^DEB.MATRIX		003314	~xDERVX	457
00D007	^DEB.MATRIXTYPE		00F314	~xDESOLVE	457
3B670	xDEC	456	07E007	^DESOLVE	391
0B1006	^DeCntMulti	375	3B1BA	xDET	457
2F32A	DECODE		3EB84	xDETACH	457
2A893	Decomp#Disp	52	2F7006	^deuxipi	420
2A878	Decomp#Line	52	03B0AB	$xDIAG \rightarrow$	457
25EAA	DECOMP\$	53	00C0DE	xDIAGMAP	
2F1BF	Decomp%Short	54	36E93	dIDNTNcase	145
2A842	Decomp1Line	52	084314	~xDIFF	457
2A8AE	DecompEcho	54	00E0AB	xDIFFEQ	457
2A85D	DecompEdit	53	16E006	^DIMLIMITS	64
2A8C9	DecompStd1Line	52	345006	^DIMRANM	337
2A8E4	DecompStd1Line32	52	004007	^DIMS	
3E576	xDECR	456	38BAE	xDIR	
0370DE	xDEDICACE	3	39725	xDISP	457
041ED	DEEPSLEEP	178	25EBC	Disp5x7	282
0250DE	xDEF	2.0	25FB8	DISP@01	281
3E85C	xDEFINE	456	25FBD	DISP@09	281
			1		- <del>-</del>

Addr.	Name	Page	Addr.	Name	Page
25FC2	DISP@17	281	220006	^DIVIS	415
25FC7	DISP@25	281	044314	~xDIVIS	457
25F16	DISP_LINE		27A006	^DIVISext	381
2EF6F	DispBadToken	51	3F4006	^DIVISIBLE?	357
2EF71	DispBadToken2		3F1006	^DIVMETAOBJ	88,361
2F19E	DispCommandLine	312,274	071314	~xDIVMOD	457
2EEFF	DispCoord1	282	249006	^DIVMOD	
2F32B	DISPCOORD2	282	3F0006	^DIVOBJext	420
2EE5A	DispEditLine	274	062314	~xDIVPC	457
2F300	DispILPrompt	274	48D006	^DIVPC!	
25FE5	DISPLASTROW	282	36E43	dLISTcase	145
25FEA	DISPLASTROWBUT1	282	3816B	xDO	457
0C0007	^DISPLAYext	98	073F7	DO	151
2DFE0	DispMenu	292,274	25EAE	DO#EXIT	156
2DFF4	DispMenu.1	292,274	25EAF	DO\$EXIT	156
25FB3	DISPN	281	25EB0	DO%EXIT	156
25FB8	DISPROW1	281	25EBE	Do1st/2nd+:	278
25EAB	DISPROW1*	281	2F2F0	DO <del< td=""><td>303</td></del<>	303
25FE0	DISPROW10	281	2F2EE	DO <skip< td=""><td>305</td></skip<>	305
25FBD	DISPROW2	281	2F2E4	DO>BEG	305
25EAC	DISPROW2*	281	2F2F1	DO>Del	303
25FC2	DISPROW3	281	2F2E5	DO>END	306
25FC7	DISPROW4	281	2EF04	DO>LCD	92
25FCC	DISPROW5	281	2F2EF	DO>Skip	305
261F7	DISPROW6	281	25EB1	DO>STR	53
25FD1	DISPROW7	281	1A7006	^DO>STRID	54
25FD6	DISPROW8	281	02E002	^DoAlert	282
25FDB	DISPROW9	281	25EBF	DoBadKey	207
38C00	DISPST2&FREEZE	282	2EEC1	DOBAUD	180
2C305	DispStatus	274	25EB2	DOBEEP	178
25EAD	DISPSTATUS2	282	2EFA6	DOBIN	177
2C2F9	DispStsBound	274	074E4	DOBIND	116
2EE5B	DispTime?		2EEC5	DOBUFLEN	180
2A7F7	DispTimeReq?	274	2EF02	DOC>PX	318
25EBD	DispVarsUtil		25EC0	DoCAlarmKey	
0160DD	xDISPXY	457	25EB3	DOCHR	46
255B5	Distance	95	09E002	^DoCKeyCancel	244
108006	^DISTDIVext	372	0A0002	^DoCKeyChAll	244
0190DE	xDISTRIB		09F002	^DoCKeyCheck	244
118007	^DISTRIB*	368	09D002	^DoCKeyOK	244
3C2006	^DISTRIB/	368	0В0002	^DoCKeyUnChAll	244
2F21A	Dither	95	2EF05	DOCLLCD	277
056314	~xDIV	457	2EECB	DOCR	181
026314	~xDIV2	457	2EECA	docr	101
31994	DIV2		2EFA8	DODEC	177
3EF006	^DIV2LISText	372	2F2F9	DODEL.L	303
072314	~xDIV2MOD	457	2EECD	DODELAY	181
22F006	^DIVERGENCE	415	25EC1	DoDelim	302
221.000	DIVENCE	410	1 20501	DODCIIII	502

Addr.	Name	Page	Addr.	Name	Page
25EC2	DoDelims	302	2EECC	DOPRLCD	
25EB4	DODISP	281	2EF01	DOPX>C	318
2604E	DOENG	177	30A66	DORANDOMIZE	33
2EEF9	DOERASE	273	25EB5	DORCLE	318
39527	xDOERR	457	2F2E9	DOREPL	308
2F2DD	DoFarBS	303	2F2EB	DOREPLACE	308
2F2DE	DoFarDel	303	2F2EC	DOREPLACE/NEXT	308
2F2E8	DOFIND	308	2AC0E	DoRunSafe	177
2EEBD	DOFINISH	180	2EEC6	DOSBRK	180
25EC3	DoFirstRow	294	26058	DOSCI	177
26053	DOFIX	177	25ECC	DoSolvrMenu	
2EF60	DOGRAPHIC	318	2EEC7	DOSRECV	180
25EC4	DoHere:	168	2605D	DOSTD	177
2EFA5	DOHEX	177	2565A	DoStdKeys?	224
2C371	DoInputForm	257	3B76C	DOSTOALLF2	176
2EEC4	DOKERRM	180	25EB6	DOSTOE	318
0AF002	^DoKeyCancel		2F23E	DOSTOSYSF	175
0B5002	^DoKeyEdit		25EB7	DOSTR>	50
25EC5	DoKeyOb	206	2EFAA	dostws	57
0B4002	^DoKeyOK		0540AB	xDOSUBS	458
25886	DoLabel	293, 95	3B1E1	xDOT	458
2EF03	DOLCD>	92	2F2DB	DOTEXTINFO	312
05B0AB	xDOLIST	457	2EEC3	DOTRANSIO	180
0B2006	^DoLS	375	25EB8	DOTVARS%	168
0210DE	*DOMAIN		0BD002	^DOTVARS{}	169
25EC6	DoMenuKey	291	25EB9	DOVARS	168
2589A	DoMenuKeyNS	293	2F2D4	dowait	172
026FE	DOMINIFONT		2EE60	DoWarning	282
0AE002	^DoMKeyOK		25EBA	DPRADIX?	178
0000B1	~DoMsgBox		3C484	xDRAW	458
25EC7	DoNameKeyLRS		06B0AB	xDRAW3DMATRIX	458
25EC8	DoNameKeyRS		2F32C	DRAWBOX#	92
2F192	DoNewEqw		2F13F	DRAWLINE#3	92
2F142	DoNewMatrix		2F32D	drax	
008007	^DoNewMatrixCplx		3C4BA	xDRAX	458
007007	^DoNewMatrixReal		36EA7	dREALNcase	145
00B007	^DoNewMatrixRealOrCplx		0230DE	xDROITE	
2F2EA	DONEXT	308	3DC3B	xDROP	458
25EC9	DoNextRow		03244	DROP	106
2EFA7	DOOCT	177	3683D	DROP#1-	24
2F13C	DoOldMatrix		282CC	DROP%0	31
00A007	^DoOldMatrixCplx		36996	DROP'	130
009007	^DoOldMatrixReal		3DC56	xDROP2	458
2EEC0	DOOPENIO		2F1C6	DROP3PICK	109
2EEC2	DOPARITY	180	28ACE	DROP?symcomp	86
2EEBE	DOPKT	180	25ECD	DropBadKey	207
25ECA	DoPlotMenu		2F32E	DROPDEADTRUE	130
25ECB	DoPrevRow		357CE	DROPDUP	106

Addr.	Name	Page	Addr.	Name	Page
35289	DROPFALSE	136	3696E	DUP'	130
364AF	DROPLOOP	151	36513	DUP1LAMBIND	116
3DCC7	xDROPN	458	3DC05	xDUP2	458
37032	DROPNDROP	75, 106	35D30	DUP3PICK	106, 106
04D3E	DROPNULL\$	46	3674D	DUP3PICK#+	24
3596D	DROPONE	21	34797	DUP4PUTLAM	119
3606B	DROPOVER	106	3432C	DUP4UNROLL	106
36342	DROPRDROP	129	35C2C	DUP@	166
36007	DROPROT	106	37258	DupAndThen	
35733	DROPSWAP	106	159006	^DUPCKLEN{}	72
3574D	DROPSWAPDROP	106, 107	3F29A	xDUPDUP	458
2F32F	DropSysErr\$		35CE0	DUPDUP	106
26062	DropSys0bs		36635	DUPEQ:	137
35280	DROPTRUE	136	25EBB	DUPGROBDIM	90
26215	DropVStack	162	3622A	DUPINCOMP	71
2E0006	^DROPZ0	327	3645A	DUPINDEX@	152
2DF006	^DROPZ1	327	357E2	DUPLEN\$	47
3558C	DROPZERO	21	3627A	DUPLENCOMP	68
3EFEF	xDTAG	458	3DCE2	xDUPN	458
35136	DTYPEARRY?	199	36252	DUPNULL\$?	55
35172	DTYPECOL?	200	16C006	^DUPNULL[]?	339
35109	DTYPECSTR?	199	36266	DUPNULLCOMP?	68
115007	^DTYPEGAUSSINT?	200,333	36ABD	DUPNULL{}?	72
168006	^DTYPEIRRQ?	326,421	36AEA	DUPONE	21
35190	DTYPELIST?	199	3611F	DUPPICK	106
170006	^DTYPENDO?	339	0F7006	^DupQIsZero?	334
35118	DTYPEREAL?	199	36133	DUPROLL	106
171006	^DTYPFMAT?	201	34FC0	DUPROM-WORD?	100
34EBE	DUMP	117	35C40	DUPROMPTR@	100
3DBEA	xDUP	458	35FF3	DUPROT	106
03188	DUP	106	35A56	DUPSAFE@	166
3532B	DUP#0<>	26	347AB	DUPTEMPENV	119
36441	DUP#0<>WHILE	151	36B26	DUPTWO	21
352BD	DUP#0=	25	350CD	DUPTYPEAPLET?	200
348F7	DUP#0=case	141	35136	DUPTYPEARRY?	199
36D21	DUP#0=csDROP	142	350EB	DUPTYPEBINT?	200
3490E	DUP#0=csedrp	141	35037	DUPTYPECHAR?	200
36ED4	DUP#0=IT	141	35127	DUPTYPECMP?	199
36F51	DUP#0=ITE	141	35172	DUPTYPECOL?	200
364C8	DUP#0_DO	151	35109	DUPTYPECSTR?	199
35912	DUP#1+	24	351AE	DUPTYPEEXT?	200
34431	DUP#1+PICK	77, 106	35082	DUPTYPEFLASHPTR?	200
35956	DUP#1-	24	350BE	DUPTYPEFONT?	200
3531C	DUP#1=	26	116007	^DUPTYPEGAUSSINT?	201,333
3571E	DUP#2+	24	35181	DUPTYPEGROB?	199
366D0	DUP#<7	26	350FA	DUPTYPEHSTR?	199
362DE	DUP\$>ID	116	35046	DUPTYPEIDNT?	199
36C0E	DUP%0=	35	350DC	DUPTYPELAM?	199

Addr.	Name	Page	Addr.	Name	Page
35190	DUPTYPELIST?	199	33157	EIGHT	10
350AF	DUPTYPELNGCMP?	200	331BB	EIGHTEEN	11
350A0	DUPTYPELNGREAL?	200	342BB	EIGHTROLL	108
35118	DUPTYPEREAL?	199	33427	EIGHTY	14
35145	DUPTYPEROMP?	200	33431	EIGHTYONE	14
35154	DUPTYPERRP?	200	33175	ELEVEN	10
0CC006	^DupTypeS?	335	4F9006	^ELMGext	399
35163	DUPTYPESYMB?	199	3805D	xELSE	458
3519F	DUPTYPETAG?	200	371B3	Embedded?	69
183006	^DUPTYPEZ?	200	2F330	ENCODE	
35091	DUPTYPEZINT?	200	2F331	ENCODE1PKT	
3457F	DUPUNROT	106, 107	38A54	xENDDO	458
36AD6	DUPZERO	21	0570AB	xENDSUB	458
10A006	^DupZIsEven?	334	38A14	xENDTIC	
0FB006	^DupZIsNeq?	334	3B5DA	xENG	458
0F9006	^DupZIsOne?	334	088314	~xEPSX0	458
109006	^DupZIsTwo?	334	03B2E	EQ	137
36A77	dvarlsBIND	117	3663A	EQ:	137
00003	DZP		3BDE6	xEQ>	458
18C006	^E%%>C%%	37	34999	EQcase	144
2C121	easyabs		34920	EQcasedrop	144
2EEE4	Echo\$Key	302	2F332	EQCURSOR?	
2F11C	Echo\$NoChr00	302	36EBB	EQIT	144
25ED1	Echo2Macros		36F01	EQITE	144
2EEE3	EchoChrKey		25ED3	EqList?	73
039EF	ECUSER		37829	EQLookup	70
25F10	ederr	158	3664E	EQOR	137
0070DD	XEDIT	458	3607F	EQOVER	137
0090DD	xEDITB	458	03B97	EQUAL	137
25ECE	EDITDECOMP\$	53	36D62	EQUALcase	144
25F11	editdecomp\$w	53	2AD81	EQUALcasedrop	145
36E57	EditExstCase	146	36D08	EQUALcasedrp	144
2F1A9	EDITF		3660D	EQUALNOT	137
2F1A8	EditFont		36E7F	EQUALNOTcase	144
2EEE5	EditLevel1	311	36662	EOUALOR	137
257A2	EditLExists?	300	3C0006	^EQUALPOS2META	78
806FD	EDITLINE		3776B	EQUALPOSCOMP	69
2EEEB	EDITLINE\$	300	3BF006	^EQUALPOSMETA	78
25ED2	EditMenu	311	25ECF	EQUATION	318
2EEEC	EDITPARTS	311	462006	^EQUATION?	86
2F2DF	EditSelect	309	4BF006	^EOUIV!	387
2EEE9	EditString	310	00B0DD	xEQW	458
02E314	~xEGCD	458	010004	^EQW3	100
3D8006	^EGCDext	380	01D004	^EOW3Code	
3DB000	^EGCDNEWG	500	01C004	^EQW3Code ^EQW3CursorOff	
3DA006	^EGCDSWAP		01B004	^EOW3CursorOn	
02C0AB	xEGV	458	011004	^EQW3Edit	311
02C0AB	xEGVL	458	019004	^EOW3GROB	98
	AUGVI	400	01/004	EČM 2GVOD	<b>J</b> O

Addr.	Name	Page	Addr.	Name	Page
01F004	^EQW3GROBmini	98	25A006	^EvalNoCKx/	417
01A004	^EQW3GROBStk	98	25B006	^EvalNoCKx^	417
01E004	^EQW3GROBsys	98	262006	^EvalNoCKxAND	418
012004	^EQW3StartEdit		25C006	^EvalNoCKxCHS	417
016004	^EQW3ViewLeft		260006	^EvalNoCKxCOMB	418
014004	^EQW3ViewLeftX		25D006	^EvalNoCKxINV	417
013004	^EQW3ViewMargin		25E006	^EvalNoCKxMOD	417
017004	^EQW3ViewRight		261006	^EvalNoCKxOR	418
018004	^EQW3ViewRightRPL		25F006	^EvalNoCKxPERM	418
015004	^EQW3ViewRightX		263006	^EvalNoCKxXOR	418
01F100	xER	479	264006	^EvalNoCKxXROOT	418
57E006	^ERABLEERROR	403	2EF69	EvalParsed	
3C553	xERASE	458	2A3006	^EVALUSERFCN	354
26071	ERASE&LEFT\$3x5	97	283006	^EVIDENText	382
2606C	ERASE&LEFT\$5x7	97	284006	^EVIDSOLV	382
3376F	Err#Kill	17	0A2007	^EXACT?	406
33819	Err#NoLstArg	17	09E007	^EXACTMODE	406
33779	Err#NoLstStk	17	098007	^EXACTOFF	406
58E006	^ERR\$EVALext	404	097007	^EXACTON	406
3955B	xERR0	458	2F333	EXCHINITPK	
092006	^ErrBadDim	403	2F2F8	EXEC_CMD	309
26067	ERRBEEP	156	258EF	ExitAction!	291
090006	^ErrInfRes	403	3709В	ExitAtLOOP	152
04ED1	ERRJMP	156	27C33	ExitFcn	
39591	xERRM	458	04E37	EXITMSGSTO	156
39576	xERRN	458	240006	^EXLR	356
04CE6	ERROR@	156	06C314	~xEXLR	459
04D33	ERRORCLR	156	3A9B7	xEXP	459
2F1A1	ErrorHandled?		087314	~xEXP&LN	
36883	ERROROUT	156	01A0DE	xEXP2POW	
04D0E	ERRORSTO	156	458006	^exp2sincos	367
04E5E	ERRSET	157	250006	^EXPAMOD	
38ABA	XERRTHEN		3E5E9	xEXPAN	459
04EB8	ERRTRAP	157	34C82	EXPAND	48,57
091006	^ErrUndefRes	403	000314	~xEXPAND	459
214006	^EULER	414	12C006	^EXPAND^	347
038314	~xEULER	458	1D7006	^EXPANDBOTH	
06F8E	EVAL	128	525006	^EXPANDLN	349
395AC	xEVAL	458	076314	~xEXPANDMOD	459
2F2E3	EVAL.LINE	309	30F006	^EXPEXPA	355
2F2FB	EVAL.SELECTION	309	313006	^EXPEXPA*	369
38C2C	xEVAL>		315006	^EXPEXPA*1	369
25ED0	EVALCRUNCH		311006	^EXPEXPA+	369
26319	EvalNoCK	198	312006	^EXPEXPA-	369
25F29	EvalNoCK:	198	314006	^EXPEXPANEG	369
257006	^EvalNoCKx*	417	529006	^xEXPext	349
258006	^EvalNoCKx+	417	3E25E	XEXPFIT	459
259006	^EvalNoCKx-	417	017314	~xEXPLN	459
			1	·	100

ABBOOF	Addr.	Name	Page	Addr.	Name	Page
A1D006	418006	^EXPLNext	356	3E1006	^FHORNER	378
33193   EXT	3AB6F	xEXPM	459	3319D	FIFTEEN	11
D5481 EXTN	41D006	^EXPM2EXP	351	332FB	FIFTY	13
2F334	33193	EXT	11	3334B	FIFTYEIGHT	13
3398B	05481	EXTN	81, 71	3332D	FIFTYFIVE	13
336F7	2F334	Extobcode		33323	FIFTYFOUR	13
33701	3398B	EXTOBOB	20	33355	FIFTYNINE	13
0050AB	336F7	EXTREAL	16	33305	FIFTYONE	13
0620AB	33701	EXTSYM	16	33341	FIFTYSEVEN	13
3ABAF	0050AB	xEYEPT	459	33337	FIFTYSIX	13
278006         ^FACTlext         381         067004         ^Filer         188           276006         ^FACTOEXT         381         00CDD         xFILER         459           296006         ^FACTOOBJEXT         384         06D004         ^FILER_MANAGER         188           28C006         ^FACTORACEXT         383         25F2C         FindlstT.1         FindlstTrue         70           28C006         ^FACTORACEXT         383         25F2C         FindlstTrue         70           077314         ^FACTORMOD         459         391AE         xFINDALARM         459           21F006         ^FACTORS         415         42006         ^FindCurVar         387           043314         ^XFACTORS         459         35A006         ^FINDELN         342           573006         ^facts         350         2F336         FindMext         576006         ^facts         350         2F336         FindELN         308           2789B         failed         136         2E712         FindDVARS         86           03AC0         FALSE         136         3ED76         XFINISH         459           36554         FALSETY         136         344F         FIVEFO	0620AB	xF0 $\lambda$	459	33319	FIFTYTHREE	13
27C006	3ABAF	xFACT	475	3330F	FIFTYTWO	13
296006	27B006	^FACT1ext	381	067004	^Filer	188
001314         *xfactor         459         06E004         *filer_Managertype         188           28C006         *factorial         383         25F2C         Find1stT.1         *foctorial         70           572006         *factorial         350         37798         Find1stTrue         70           077314         *xfactormod         459         391AE         xfINDALARM         459           21F006         *factors         415         4C2006         *findCurVar         387           043314         *xfactors         459         35A006         *fINDELN         342           573006         *facts         350         2F336         Findmext           576006         *factsint         330         2F2F2         FindStrInCmd         308           27E9B         failed         136         2F110         FINDVARS         86           03ACO         FALSE         136         3ED76         xFINISH         459           369FF         FALSE         136         3ED76         xFINISH         459           36554         FalseFalse         136         36C4C         FIRSTC+         300           3654         FALSETRUE         136         3344F <td>27C006</td> <td>^FACTOext</td> <td>381</td> <td>00C0DD</td> <td>xFILER</td> <td>459</td>	27C006	^FACTOext	381	00C0DD	xFILER	459
28C006	296006	^FACTOOBJext	384	06D004	^FILER_MANAGER	188
572006         ^factorial         350         37798         Find1stTrue         70           077314         ~xFACTORMOD         459         391AE         xFINDALARM         459           21F006         ^FACTORS         415         4C2006         ^FindCurVar         387           043314         ~xFACTORS         459         35A006         ^FINDELN         342           573006         ^facts         350         2F336         FindNext           576006         ^factzint         330         2F2F2         FindStrInCmd         308           27E9B         failed         136         2F110         FINDVARS         86           03AC0         FALSE         136         3ED76         xFINISH         459           369FF         FALSE'         130         264DB         FIRSTC+         459           369FF         FALSE'         130         264DB         FIRSTC+         300           3654         FalseFalse         136         264CC         FIRSTC@         300           3654         FALSETRUE         136         3344F         FIVEFOUR         14           0600AB         xFANNING         459         34257         FIVEROLL         108 <td>001314</td> <td>~xFACTOR</td> <td>459</td> <td>06E004</td> <td>^FILER_MANAGERTYPE</td> <td>188</td>	001314	~xFACTOR	459	06E004	^FILER_MANAGERTYPE	188
077314         ~xFACTORMOD         459         391AE         xFINDALARM         459           21F006         ^FACTORS         415         4C2006         ^FindCurVar         387           043314         ~xFACTORS         459         35A006         ^FINDELN         342           573006         ^facts         350         2F336         FindNext           576006         ^facts         330         2F2F2         FindStrInCmd         308           27E9B         failed         136         2F110         FINDVARS         86           03AC0         FALSE         136         3ED76         xFINISH         459           369FF         FALSE         130         264DB         FIRSTC+         100           369FF         FALSE         136         3ED76         xFINISH         459           369FF         FALSE         130         264DB         FIRSTC+         100           369FF         FALSE         136         3ED76         xFINISH         459           369FF         FALSE         130         264DB         FIRSTC+         100           36554         FALSETRUE         136         33139         FIVE         10	28C006	^FACTORACext	383	25F2C	Find1stT.1	
21F006         ^FACTORS         415         4C2006         ^FindCurVar         387           043314         ~XFACTORS         459         35A006         ^FINDELN         342           573006         ^facts         350         2F336         FindNext           576006         ^factzint         330         2F2F2         FindStrInCmd         308           27E9B         failed         136         2F110         FINDVARS         86           03AC0         FALSE         136         3ED76         xFINISH         459           369FF         FALSE'         130         264DB         FIRSTC+         459           283E8         FalseFalse         136         264CC         FIRSTC#         300           36554         FalseTrue         136         33139         FIVE         10           36554         FALSETRUE         136         3344F         FIVEFOUR         14           0600AB         xFANNING         459         34257         FIVEROLL         108           3F2DF         xFAST3D         459         33463         FIVESIX         14           09A006         ^FASTREDUCE         384         34357         FIVEUNCL         109 </td <td>572006</td> <td>^factorial</td> <td>350</td> <td>37798</td> <td>Find1stTrue</td> <td>70</td>	572006	^factorial	350	37798	Find1stTrue	70
043314	077314	~xFACTORMOD	459	391AE	xFINDALARM	459
573006       ^facts       350       2F336       FindNext         576006       ^factzint       330       2F2F2       FindStrInCmd       308         27E9B       failed       136       2F110       FINDVARS       86         03AC0       FALSE       136       3ED76       xFINISH       459         369FF       FALSE'       130       264DB       FIRSTC+         283E8       FalseFalse       136       264CC       FIRSTC⊕       300         36554       FalseTrue       136       33139       FIVE       10         36554       FALSETRUE       136       3344F       FIVEFOUR       14         0600AB       xFANNING       459       34257       FIVEROLL       108         3F2DF       xFAST3D       459       33463       FIVESIX       14         3F6006       ^FastDiv?       357       33445       FIVETHREE       14         09A006       ^FASTREDUCE       384       34357       FIVEUNROLL       109         255DD       FBOXB       94       2F337       FixRPP         255D8       FBOXG2       94       105007       ^fk+1/fk         255D2       FBOXXOr <t< td=""><td>21F006</td><td>^FACTORS</td><td>415</td><td>4C2006</td><td>^FindCurVar</td><td>387</td></t<>	21F006	^FACTORS	415	4C2006	^FindCurVar	387
576006         ^factzint         330         2F2F2         FindstrInCmd         308           27E9B         failed         136         2F110         FINDVARS         86           03AC0         FALSE         136         3ED76         xFINISH         459           369FF         FALSE'         130         264DB         FIRSTC+         300           283E8         FalseFalse         136         264CC         FIRSTC@         300           36554         FalseTrue         136         33139         FIVE         10           36554         FALSETRUE         136         3344F         FIVEFOUR         14           0600AB         xFANNING         459         34257         FIVEROLL         108           3F2DF         xFAST3D         459         33463         FIVESIX         14           3F6006         ^FASTREDUCE         384         34357         FIVEURREE         14           09A006         ^FASTREDUCE         384         34357         FIVEURREE         14           255DD         FBOXB         94         2F337         FixRRP           255D8         FBOXG2         94         105007         ^fk+1/fk           255D3 </td <td>043314</td> <td>~xFACTORS</td> <td>459</td> <td>35A006</td> <td>^FINDELN</td> <td>342</td>	043314	~xFACTORS	459	35A006	^FINDELN	342
27E9B       failed       136       2F110       FINDVARS       86         03AC0       FALSE       136       3ED76       xFINISH       459         369FF       FALSE'       130       264DB       FIRSTC+         283E8       FalseFalse       136       264CC       FIRSTC@       300         36554       FalseTrue       136       33139       FIVE       10         36554       FALSETRUE       136       3344F       FIVEFOUR       14         0600AB       xFANNING       459       34257       FIVEROLL       108         3F2DF       xFAST3D       459       33463       FIVESIX       14         3F6006       ^FastDiv?       357       33445       FIVETHREE       14         09A006       ^FASTREDUCE       384       34357       FIVEUNROLL       109         255DD       FBOXB       94       3B59A       xFIX       459         255D3       FBOXG1       94       2F337       FixRPP         255D3       FBOXG2       94       105007       ^fk+1/fk         255D3       FBOXW       94       201006       ^fLAGACINZC       413         3B635       xFC?	573006	^facts	350	2F336	FindNext	
03ACO       FALSE       136       3ED76       xFINISH       459         369FF       FALSE'       130       264DB       FIRSTC+         283E8       FalseFalse       136       264CC       FIRSTC@       300         36554       FalseTrue       136       33139       FIVE       10         36554       FALSETRUE       136       3344F       FIVEFOUR       14         0600AB       xFANNING       459       34257       FIVEROLL       108         3F2DF       xFAST3D       459       33463       FIVESIX       14         3F6006       ^FastDiv?       357       33445       FIVETHREE       14         09A006       ^FASTREDUCE       384       34357       FIVEUNROLL       109         255DD       FBOXB       94       3B59A       xFIX       459         255D3       FBOXG1       94       2F337       FixRRP         255D8       FBOXG2       94       105007       ^fk+1/fk         255D2       FBOXXOr       94       201006       ^FLAGASIN2C       413         3B529       xFC?       459       1FF006       ^FLAGASIN2C       413         3B635       xFC?C <td>576006</td> <td>^factzint</td> <td>330</td> <td>2F2F2</td> <td>FindStrInCmd</td> <td>308</td>	576006	^factzint	330	2F2F2	FindStrInCmd	308
369FF       FALSE'       130       264DB       FIRSTC+         283E8       FalseFalse       136       264CC       FIRSTC@       300         36554       FalseTrue       136       33139       FIVE       10         36554       FALSETRUE       136       3344F       FIVEFOUR       14         0600AB       xFANNING       459       34257       FIVEROLL       108         3F2DF       xFAST3D       459       33463       FIVESIX       14         3F6006       ^FastDiv?       357       33445       FIVETHREE       14         09A006       ^FASTREDUCE       384       34357       FIVEUNROLL       109         255DD       FBOXB       94       3B59A       xFIX       459         255D3       FBOXG1       94       2F337       FixRRP         255D8       FBOXG2       94       105007       ^fk+1/fk         255D3       FBOXW       94       201006       ^FLAGACOS2S       413         38529       xFC?       459       1FF006       ^FLAGASIN2C       413         3B635       xFC?C       459       1FE006       ^FLAGATAN2S       413         3D81D       xFCNAPP	27E9B	failed	136	2F110	FINDVARS	86
283E8       FalseFalse       136       264CC       FIRSTC@       300         36554       FalseTrue       136       33139       FIVE       10         36554       FALSETRUE       136       3344F       FIVEFOUR       14         0600AB       xFANNING       459       34257       FIVEROLL       108         3F2DF       xFAST3D       459       33463       FIVESIX       14         3F6006       ^FastDiv?       357       33445       FIVETHREE       14         09A006       ^FASTREDUCE       384       34357       FIVEUNROLL       109         255DD       FBoxB       94       3B59A       xFIX       459         255D3       FBoxG1       94       2F337       FixRRP         255D8       FBoxG2       94       105007       ^fk+1/fk         255D3       FBoxW       94       201006       ^FLAGACOS2S       413         255E2       FBoxXor       94       200006       ^FLAGASIN2C       413         3B529       xFC?       459       1FE006       ^FLAGATAN2S       413         3B635       xFC?C       459       1FE006       ^FLAGACHINREM       414         21	03AC0	FALSE	136	3ED76	xFINISH	459
36554       FalseTrue       136       33139       FIVE       10         36554       FALSETRUE       136       3344F       FIVEFOUR       14         0600AB       xFANNING       459       34257       FIVEROLL       108         3F2DF       xFAST3D       459       33463       FIVESIX       14         3F6006       ^FASTREDUCE       384       34357       FIVEUNROLL       109         255DD       FBOXB       94       3B59A       xFIX       459         255D3       FBOXG1       94       2F337       FixRRP         255D8       FBOXG2       94       105007       ^fk+1/fk         255D3       FBOXW       94       201006       ^FLAGACOS2S       413         255D2       FBOXXOr       94       200006       ^FLAGASIN2C       413         3B529       xFC?       459       1FF006       ^FLAGASIN2C       413         3B635       xFC?C       459       1FE006       ^FLAGAXQ       346         2F335       FCNUtilEnd       216006       ^FLAGCHINREM       414         210006       ^FLAGDERIV       412         041314       ~xFCOEF       459       1EC006       <	369FF	FALSE'	130	264DB	FIRSTC+	
36554       FALSETRUE       136       3344F       FIVEFOUR       14         0600AB       xFANNING       459       34257       FIVEROLL       108         3F2DF       xFAST3D       459       33463       FIVESIX       14         3F6006       ^FASTDIV?       357       33445       FIVETHREE       14         09A006       ^FASTREDUCE       384       34357       FIVEUNROLL       109         255DD       FBOXB       94       3B59A       xFIX       459         255D3       FBOXG1       94       2F337       FixRRP         255D8       FBOXG2       94       105007       ^fk+1/fk         255D3       FBOXW       94       201006       ^FLAGACOS2S       413         255D2       FBOXXOr       94       200006       ^FLAGASIN2C       413         3B529       xFC?       459       1F006       ^FLAGASIN2T       413         3B635       xFC?C       459       1FE006       ^FLAGAXQ       346         2F335       FcnUtilend       216006       ^FLAGCHINREM       414         210006       ^FLAGDERIV       412         041314       ~xFCOEF       459       1EC006	283E8	FalseFalse	136	264CC	FIRSTC@	300
0600AB       xFANNING       459       34257       FIVEROLL       108         3F2DF       xFAST3D       459       33463       FIVESIX       14         3F6006       ^FASTDIV?       357       33445       FIVETHREE       14         09A006       ^FASTREDUCE       384       34357       FIVEUNROLL       109         255DD       FBOXB       94       3B59A       xFIX       459         255D3       FBOXG1       94       2F337       FixRRP         255D8       FBOXG2       94       105007       ^fk+1/fk         255D3       FBOXW       94       201006       ^FLAGACOS2S       413         255E2       FBOXXOr       94       200006       ^FLAGASIN2C       413         3B529       xFC?       459       1FF006       ^FLAGASIN2T       413         3B635       xFC?C       459       1FE006       ^FLAGAXQ       346         2F335       FCNUtilEnd       225006       ^FLAGCHINREM       414         210006       ^FLAGDERIV       412         041314       ~xFCOEF       459       1EC006       ^FLAGDESOLVE       412         289006       ^FDEG2ext       382       207006	36554	FalseTrue	136	33139	FIVE	10
3F2DF       xFAST3D       459       33463       FIVESIX       14         3F6006       ^FastDiv?       357       33445       FIVETHREE       14         09A006       ^FASTREDUCE       384       34357       FIVEUNROLL       109         255DD       FBoxB       94       3B59A       xFIX       459         255D3       FBoxG1       94       2F337       FixRPP         255D8       FBoxG2       94       105007       ^fk+1/fk         255D3       FBoxW       94       201006       ^FLAGACOS2S       413         255E2       FBoxXor       94       200006       ^FLAGASIN2C       413         3B529       xFC?       459       1FF006       ^FLAGASIN2T       413         3B635       xFC?C       459       1FE006       ^FLAGATAN2S       413         3D81D       xFCNAPPLY       225006       ^FLAGCHINREM       414         210006       ^FCOEF       415       1E9006       ^FLAGDERIV       412         041314       ~xFCOEF       459       1EC006       ^FLAGDESOLVE       412         289006       ^FDEG2ext       382       207006       ^FLAGDIV2       413	36554	FALSETRUE	136	3344F	FIVEFOUR	14
3F6006       ^FastDiv?       357       33445       FIVETHREE       14         09A006       ^FASTREDUCE       384       34357       FIVEUNROLL       109         255DD       FBoxB       94       3B59A       xFIX       459         255D3       FBoxG1       94       2F337       FixRRP         255D8       FBoxG2       94       105007       ^fk+1/fk         255D3       FBoxW       94       201006       ^fLAGACOS2S       413         255E2       FBoxXor       94       200006       ^fLAGASIN2C       413         3B529       xFC?       459       1FF006       ^fLAGASIN2T       413         3B635       xFC?C       459       1FE006       ^fLAGATAN2S       413         3D81D       xFCNAPPLY       225006       ^fLAGAXQ       346         2F335       FcnUtilend       216006       ^fLAGCHINREM       414         210006       ^fCOEF       415       1E9006       ^fLAGDERIV       412         041314       ~xFCOEF       459       1EC006       ^fLAGDESOLVE       412         289006       ^fDEG2ext       382       207006       ^fLAGDIV2       413	0600AB	xFANNING	459	34257	FIVEROLL	108
09A006         ^FASTREDUCE         384         34357         FIVEUNROLL         109           255DD         FBoxB         94         3B59A         xFIX         459           255D3         FBoxG1         94         2F337         FixRRP           255D8         FBoxG2         94         105007         ^fk+1/fk           255D3         FBoxW         94         201006         ^fLAGACOS2S         413           255E2         FBoxXor         94         200006         ^fLAGASIN2C         413           3B529         xFC?         459         1FF006         ^fLAGASIN2T         413           3B635         xFC?C         459         1FE006         ^fLAGATAN2S         413           3D81D         xFCNAPPLY         225006         ^fLAGAXQ         346           2F335         FcnUtilend         216006         ^fLAGCHINREM         414           210006         ^fCOEF         415         1E9006         ^fLAGDERIV         412           041314         ~xFCOEF         459         1EC006         ^fLAGDESOLVE         412           289006         ^fDEG2ext         382         207006         ^fLAGDIV2         413	3F2DF	xFAST3D	459	33463	FIVESIX	14
255DD       FBoxB       94       3B59A       xFIX       459         255D3       FBoxG1       94       2F337       FixRRP         255D8       FBoxG2       94       105007       ^fk+1/fk         255D3       FBoxW       94       201006       ^fLAGACOS2S       413         255E2       FBoxXor       94       200006       ^fLAGASIN2C       413         3B529       xFC?       459       1FF006       ^fLAGASIN2T       413         3B635       xFC?C       459       1FE006       ^fLAGATAN2S       413         3D81D       xFCNAPPLY       225006       ^fLAGAXQ       346         2F335       FcnUtilend       216006       ^fLAGCHINREM       414         210006       ^fCOEF       415       1E9006       ^fLAGDERIV       412         041314       ~xFCOEF       459       1EC006       ^fLAGDESOLVE       412         289006       ^fDEG2ext       382       207006       ^fLAGDIV2       413	3F6006	^FastDiv?	357	33445	FIVETHREE	14
255D3       FBoxG1       94       2F337       FixRRP         255D8       FBoxG2       94       105007       ^fk+1/fk         255D3       FBoxW       94       201006       ^fLAGACOS2S       413         255E2       FBoxXor       94       200006       ^fLAGASIN2C       413         3B529       xFC?       459       1FF006       ^fLAGASIN2T       413         3B635       xFC?C       459       1FE006       ^fLAGATAN2S       413         3D81D       xFCNAPPLY       225006       ^fLAGAXQ       346         2F335       FcnUtilend       216006       ^fLAGCHINREM       414         21D006       ^fCOEF       415       1E9006       ^fLAGDERIV       412         041314       ~xFCOEF       459       1EC006       ^fLAGDESOLVE       412         289006       ^fDEG2ext       382       207006       ^fLAGDIV2       413	09A006	^FASTREDUCE	384	34357	FIVEUNROLL	109
255D8       FBoxG2       94       105007       ^fk+1/fk         255D3       FBoxW       94       201006       ^fLAGACOS2S       413         255E2       FBoxXor       94       200006       ^fLAGASIN2C       413         3B529       xFC?       459       1FF006       ^fLAGASIN2T       413         3B635       xFC?C       459       1FE006       ^fLAGATAN2S       413         3D81D       xFCNAPPLY       225006       ^fLAGAXQ       346         2F335       FcnUtilend       216006       ^fLAGCHINREM       414         21D006       ^fCOEF       415       1E9006       ^fLAGDERIV       412         041314       ~xFCOEF       459       1EC006       ^fLAGDESOLVE       412         289006       ^fDEG2ext       382       207006       ^fLAGDIV2       413	255DD	FBoxB	94	3B59A	xFIX	459
255D3         FBoxW         94         201006         ^FLAGACOS2S         413           255E2         FBoxXor         94         200006         ^FLAGASIN2C         413           3B529         xFC?         459         1FF006         ^FLAGASIN2T         413           3B635         xFC?C         459         1FE006         ^FLAGATAN2S         413           3D81D         xFCNAPPLY         225006         ^FLAGAXQ         346           2F335         FcnUtilend         216006         ^FLAGCHINREM         414           21D006         ^FCOEF         415         1E9006         ^FLAGDERIV         412           041314         ^xFCOEF         459         1EC006         ^FLAGDESOLVE         412           289006         ^FDEG2ext         382         207006         ^FLAGDIV2         413	255D3	FBoxG1	94	2F337	FixRRP	
255E2       FBoxXor       94       200006       ^FLAGASIN2C       413         3B529       xFC?       459       1FF006       ^FLAGASIN2T       413         3B635       xFC?C       459       1FE006       ^FLAGATAN2S       413         3D81D       xFCNAPPLY       225006       ^FLAGAXQ       346         2F335       FcnUtilend       216006       ^FLAGCHINREM       414         21D006       ^FCOEF       415       1E9006       ^FLAGDERIV       412         041314       ^xFCOEF       459       1EC006       ^FLAGDESOLVE       412         289006       ^FDEG2ext       382       207006       ^FLAGDIV2       413	255D8	FBoxG2	94	105007	^fk+1/fk	
3B529       xFC?       459       1FF006       ^FLAGASIN2T       413         3B635       xFC?C       459       1FE006       ^FLAGATAN2S       413         3D81D       xFCNAPPLY       225006       ^FLAGAXQ       346         2F335       FcnUtilend       216006       ^FLAGCHINREM       414         21D006       ^FCOEF       415       1E9006       ^FLAGDERIV       412         041314       ^xFCOEF       459       1EC006       ^FLAGDESOLVE       412         289006       ^FDEG2ext       382       207006       ^FLAGDIV2       413	255D3	FBoxW	94	201006	^FLAGACOS2S	413
3B635       xFC?C       459       1FE006       ^FLAGATAN2S       413         3D81D       xFCNAPPLY       225006       ^FLAGAXQ       346         2F335       FcnUtilend       216006       ^FLAGCHINREM       414         21D006       ^FCOEF       415       1E9006       ^FLAGDERIV       412         041314       ~xFCOEF       459       1EC006       ^FLAGDESOLVE       412         289006       ^FDEG2ext       382       207006       ^FLAGDIV2       413	255E2	FBoxXor	94	200006	^FLAGASIN2C	413
3D81D       xFCNAPPLY       225006       ^FLAGAXQ       346         2F335       FcnUtilEnd       216006       ^FLAGCHINREM       414         21D006       ^FCOEF       415       1E9006       ^FLAGDERIV       412         041314       ^xFCOEF       459       1EC006       ^FLAGDESOLVE       412         289006       ^FDEG2ext       382       207006       ^FLAGDIV2       413	3B529	xFC?	459	1FF006	^FLAGASIN2T	413
2F335       FcnUtilEnd       216006       ^FLAGCHINREM       414         21D006       ^FCOEF       415       1E9006       ^FLAGDERIV       412         041314       ~xFCOEF       459       1EC006       ^FLAGDESOLVE       412         289006       ^FDEG2ext       382       207006       ^FLAGDIV2       413	3B635	xFC?C	459	1FE006	^FLAGATAN2S	413
21D006       ^FCOEF       415       1E9006       ^FLAGDERIV       412         041314       ~xFCOEF       459       1EC006       ^FLAGDESOLVE       412         289006       ^FDEG2ext       382       207006       ^FLAGDIV2       413	3D81D	xFCNAPPLY		225006	^FLAGAXQ	346
041314 ~xFCOEF       459       1EC006 ^FLAGDESOLVE       412         289006 ^FDEG2ext       382       207006 ^FLAGDIV2       413	2F335	FcnUtilEnd		216006	^FLAGCHINREM	414
289006 ^FDEG2ext 382 207006 ^FLAGDIV2 413	21D006	^FCOEF	415	1E9006	^FLAGDERIV	412
	041314	~xFCOEF	459	1EC006	^FLAGDESOLVE	412
0100DE **EDICEDID   24D00C	289006	^FDEG2ext	382	207006	^FLAGDIV2	413
NTONDE XENTOIKIR   ZANONO ELTWAND	0180DE	xFDISTRIB		24D006	^FLAGDIV2MOD	
282006 ^FEVIDENText 377   239006 ^FLAGDIVPC 416	282006	^FEVIDENText	377	239006	^FLAGDIVPC	416
01A0AB xFFT 459 251006 ^FLAGEXPAMOD	01A0AB	XFFT	459	251006	^FLAGEXPAMOD	

Addr.	Name	Page	Addr.	Name	Page
1D6006	^FLAGEXPAND	411	1FA006	^FLAGTRIGTAN	413
1F3006	^FLAGEXPLN	413	23A006	^FLAGTRUNC	416
1D8006	^FLAGFACTOR	411	1F1006	^FLAGTSIMP	413
252006	^FLAGFACTORMOD	395	0170AB	xFLASHEVAL	459
226006	^FLAGGAUSS	346	25ED4	FlashMsg	282
208006	^FLAGGCD	413	860CC	FlashROMTAB2	
1FC006	^FLAGHALFTAN	413	2EE61	FlashWarning	282
213006	^FLAGHORNER	414	2DCB5	FLOAT	
1E5006	^FLAGIBP	412	192006	^FLOAT?	201
1DB006	^FLAGIDNTEXEC	411	3ACD1	xFLOOR	459
1EB006	^FLAGILAP	412	261CA	FLUSH	205
1DC006	^FLAGINTVX	411	261CA	FLUSHKEYS	205
22D006	^FLAGJORDAN	345	2EEC8	FLUSHRSBUF	
1EA006	^FLAGLAP	412	2F113	<pre>FNDALARM{ }</pre>	
20E006	^FLAGLCM	414	00F0DD	xFONT6	459
1EE006	^FLAGLDECSOLV		00E0DD	xFONT7	459
1ED006	^FLAGLDSSOLV	412	00D0DD	xFONT8	459
20D006	^FLAGLGCD	414	2621A	FONT>	283
1F0006	^FLAGLIN	412	0030DD	$\texttt{xFONT}{\longrightarrow}$	459
1D9006	^FLAGLISTEXEC	411	06F004	^FontBrowser	
1F2006	^FLAGLNCOLLECT	413	3DB62	xFORMUNIT	
1E0006	^FLAGMATRIXLIMIT	412	33297	FORTY	12
24F006	^FLAGMPOWMOD		332E7	FORTYEIGHT	12
094007	^FLAGNAME	405	332C9	FORTYFIVE	12
210006	^FLAGPARTFRAC	414	332BF	FORTYFOUR	12
24E006	^FLAGPOWMOD		332F1	FORTYNINE	12
1E6006	^FLAGPREVAL	412	332A1	FORTYONE	12
211006	^FLAGPROPFRAC	414	332DD	FORTYSEVEN	12
212006	^FLAGPTAYL	414	332D3	FORTYSIX	12
224006	^FLAGQXA	345	332B5	FORTYTHREE	12
10E007	^FLAGRESULTANT	380	332AB	FORTYTWO	12
1E8006	^FLAGRISCH	412	3312F	FOUR	10
1E2006	^FLAGSERIES	412	333B9	FOURFIVE	14
23B006	^FLAGSEVAL	416	05E314	~xFOURIER	459
20F006	^FLAGSIMP2	414	236006	^FOURIER	415
1F4006	^FLAGSINCOS	413	2D4006	^FOURIERext	424
0FA007	^FLAGSUM		3423A	FOURROLL	107
0FC007	^FLAGSUMVX		36043	FOURROLLROT	108
227006	^FLAGSYLVESTER	346	33193	FOURTEEN	11
1DA006	^FLAGSYMBEXEC	411	333A5	FOURTHREE	13
1FB006	^FLAGTAN2SC	413	3339B	FOURTWO	13
1FD006	^FLAGTAN2SC2	413	33297	FOURTY	12
1F6006	^FLAGTCOLLECT	413	34331	FOURUNROLL	108
1EF006	^FLAGTEXPAND	412	3AC87	xFP	459
1F5006	^FLAGTLIN	413	0D5007	^FR2ND%	425
1F7006	^FLAGTRIG	413	0CB007	^FRACPARITY	425
1F8006	^FLAGTRIGCOS	413	3EB2C	XFREE	
1F9006	^FLAGTRIGSIN	413	39745	xFREEZE	460

Addr.	Name	Page	Addr.	Name	Page
27F006	^FRND	382	31518	GETCD0	
267006	^FROMLISText	418	2F338	GetChkPRTPAR	
042314	~xFROOTS	460	04A41	GETDF	288
21E006	^FROOTS	415	26224	GetElemBotVStack	163
3B509	xFS?	460	26229	GetElemTopVStack	163
3B615	xFS?C	460	2F339	GetEqN	318
2621F	FSCANFONT		57D006	^GETERABLEMSG	403
001004	^FSTR1	54	04E07	GETEXITMSG	156
00A004	^FSTR10		0BB002	^GetFieldVals	
00B004	^FSTR11		26238	GetFontCmdHeight	
00C004	^FSTR12		2623D	GetFontHeight	
00D004	^FSTR13	55	26242	GetFontStkHeight	283
002004	^FSTR2		26247	GetHeader	273
003004	^FSTR3	54	3C22D	xGETI	460
004004	^FSTR4	54	2F106	GETINDEP	317
005004	^FSTR5	54	2EEBF	GetIOPAR	181
006004	^FSTR6	54	2F33A	GetKermPkt#	
007004	^FSTR7	55	25ED6	GETKEY	206
008004	^FSTR8		25ED7	GETKEY*	206
009004	^FSTR9	55	25ED9	GetKeyOb	206
0FE007	^FTAYL	393	2F33B	GETKP	
0AE007	^FULLDATA	407	075A5	GETLAM	117
3C955	xFUNCTION	460	3483E	GETLAMPAIR	119
06B314	~xFXND	460	2EF6D	GetLastEdit	
164006	^FXNDext	380	2F33C	getmatchtok	51
0070DE	xGAMMA	460	25EDA	GetMenu%	289
05F42	GARBAGE	178	2624C	GetMetaVStack	162
383006	^GAUSS	346	25F17	GetMetaVStackDROP	161
04D314	~xGAUSS	460	002102	XGETNAME	479
095006	^GBASIS	384	2EEBC	GETNAME	180
26076	GBUFF	272	003102	XGETNAMES	479
25ED5	GBUFFGROBDIM	273	004102	XGETNEAR	479
02C314	~xGCD	460	25EDB	GetNextToken	50
24A006	^GCD1MOD	100	2F33D	GETPARAM	316
2B4006	^GCDext		2F107	GETPMIN&MAX	317
0EC006	^GCDHEUext	399	04A0B	GETPROC	288
075314	~xGCDMOD	460	2EEF5	GETPTYPE	317
2F105	GDISPCENTER	318	2F10D	GETRES	317
3C1C7	XGET	460	2F10A	GetRes	01.
2F2F3	GET.W->	308	2F108	GETRHS	
2F2F4	GET.W<-	308	07A007	^GetRoot	419
34504	get1	77	2F33E	GETSCALE	317
2F2F6	GET_CUR_FONT.EXT	312	2F33F	GETSERIAL	011
314E4	GETAB0	012	04D64	GETTHEMESG	157
314E4 314CA	GETABO GETAB1		04714	GETTOUCH	206
001102	XGETADR	479	25967	GetUserKeys	208
001102 0371D	GETATELN	64	25967 25F18	GetVStack	161
2DDD5	getBPOFF	04	26233	GetVStackProtectWord	164
∠טטטס	gerbrorr 9		∠0∠33	GetVStackProtectWord	104

^FRND - hxs>\$ 519

ZPOFF GETXMIN	Addr.	Name	Page	Addr.	Name	Page
2P109   GETXPOS   407006	2F0FE	GETXMAX	316	009100	xH→S	478
2F109	2F0FF	GETXMIN	316	046314	~xHADAMARD	461
2F10E GETYMAX 317 3880D XHALT 461 2F100 GETYMIN 317 2608F HARDBUFF 273 2F008 getypos 26094 HARDBUFF 273 2F340 GETYPOS 25EDE HARDBUFF 273 097006 ^GPACTOR 384 2EED6 HBUFF_X_Y 273 097006 ^GPACTOR 384 2EED6 HBUFF_X_Y 273 097006 ^GPACTOR 384 05090 XHEADER→ 461 0660AB xgmol 26099 HEIGHTENGROB 273 3C74A xGOR 460 232006 ^HBERMITE 415 25588 Gor 93 05C314 *XHERMITE 461 107007 ^GOSPER? 425 059314 *XHESS 461 107007 ^GOSPER? 425 059314 *XHESS 461 3A531 GOTO 129 231006 'HBSSIAN 415 2F2E6 GOTOLABEL 306 3B68B XHEX 461 3B57F xGRAD 460 054314 *XHILBERT 461 0090DE xGRAMSCHNIDT 220006 'HILBERTNOCK 415 3C5AE xGRAPH 460 3C9C1 XHISTOGRAM 461 2F341 GraphicExit 25636 HISTON? 098006 'GREDUCE 3514C XHISTPLOT 461 33C1B xGROB 3B14C XHMS→ 461 3317F grob 11 39405 XHOME 461 3317F grob 11 39405 XHOME 461 26080 GROBIZERO 91 3E3006 'HORNAIN! 2607B GROB! 90 08D92 HOMEDIR 169 26080 GROBIZERO 91 48006 'HORNAIN! 2F3F42 GROB+# 91 48006 'HORNAIN! 2F3F42 GROB-# 91 48006 'HORNAIN! 2F3F42 GROB-# 91 48006 'HORNAIN! 2F3F42 GROB-# 91 48006 'HORNAIN! 2F3F44 GROB-# 91 48006 'HORNAIN! 2F5DB GROB-# 91 48006 'HORNAIN! 2F5DB GROB-# 91 48006 'HORNAIN! 2F5DB GROB-H 91 48006 'HORNAIN! 2F5DB GROB-H 91 48006 'HORNAIN! 2F5DB GROB-# 91 48006 'HORNAIN! 2F5DB GROB-H 91 48006 'HORNERL 424 0F007 'GROBADDEX 92 4A1006 'HORNERL 537 2F5DB GROB-HM 90 4A3006 'HORNERL 537 2F5DB GROB-HM 90 4A3006 'HORNERL 537 387 3870B A300R 461 2F0EF HXS<=HXS 558 360D006 'HSZ 328 05A03 HXSS# 558 360D006 'HSZ 328 05A03 HXSS# 25E 360D00100 XH→ 478 2EFCC HXS>\$ 46	2F007	getxpos		407006	^HALFTAN	356
2F100 GETYMIN 317 2608F HARDBUFF 273 2F008 getypos 26094 HARDBUFF 273 2F340 GETYPOS 25EDE HARDBUFF 273 097006	2F109	GETXPOS		020314	~xHALFTAN	461
2F008	2F10E	GETYMAX	317	3880D	xHALT	461
2F340 GETYPOS 384 2EDDE HARDHEIGHT 273 097006 'GFACTOR 384 2EDDE HBUFF_X_Y 279 0C8000 'GFIdVal 0050DD ×HEADER→ 461 0660AB ×gmol 26099 HEIGHTENGROB 273 094006 'GMSGLV 384 0320DE ×HELP 3C74A ×GOR 460 232006 'HERMITE 415 25588 GOT 93 05C314 "**HEENITE 461 107007 'GOSPER? 425 059314 "***HESS 461 34A31 GOTO 129 231006 'HESSIAN 415 3E57F ×GRAD 460 054314 "***HILBERT 461 3B57F ×GRAD 460 054314 "***HILBERT 461 3B57F ×GRAD 460 054314 "***HILBERT 461 0090DE ×GRAMSCHMIDT 22C006 "HILBERTNOCK 415 3C5AE ×GRAPH 460 3C9C1 ×HISTOGRAM 461 2F341 GraphicExit 25636 HISTON? 098006 'GREDUCE 3E1CA ×HISTPLOT 461 00A0AB ×GRIDMAP 460 3B14C ×HMS- 461 3317F grob 111 39405 ×HOME 461 3317F grob 111 39405 ×HOME 461 3317F grob 111 39405 ×HOME 461 2607B GROB! 90 08E92 HOMEDIR 169 26080 GROB!ZERO 91 3E3006 "HORNASIN! 12EDB GROB+ 90 4A9006 "HORNASIN! 12EDBB GROB+ 90 4A9006 "HORNASIN! 12EDBB GROB- 9	2F100	GETYMIN	317	2608F	HARDBUFF	273
097006	2F008	getypos		26094	HARDBUFF2	273
0C80B0         "gFldVal         0050DD         xHEADER→         461           0660AB         xgmol         26099         HEIGHTENGROB         273           094006         ^GMSOLV         384         0320DE         xHELP           3C74A         xGOR         460         232006         *HERMITE         415           25588         GOr         93         05C314         *xHERMITE         461           107007         ^GOSPER?         425         059314         *xHESS         461           107007         GOSPER?         425         059314         *xHESS         461           34331         GOTO         129         231006         *HESSIAN         415           25266         GOTOLABEL         306         3B68B         *HEX         461           3857F         xGRAD         460         054314         *xHLBERT         461           365AE         xGRADH         460         3C9C1         *HISTOGRAM         461           255AI         GraphicExit         25636         HISTON?         461           098006         *GREDUCE         3E1CA         *HISTPLOT         461           381CB         xGRI         MSCHIMAP <td< td=""><td>2F340</td><td>GETYPOS</td><td></td><td>25EDE</td><td>HARDHEIGHT</td><td>273</td></td<>	2F340	GETYPOS		25EDE	HARDHEIGHT	273
0660AB	097006	^GFACTOR	384	2EED6	HBUFF_X_Y	279
094006	0C80B0	~gFldVal		0050DD	$xHEADER \rightarrow$	461
3C74A	0660AB	xgmol		26099	HEIGHTENGROB	273
25588   Gor   93   05C314	094006	^GMSOLV	384	0320DE	xHELP	
107007	3C74A	xGOR	460	232006	^HERMITE	415
34A31   GOTO   129   231006	25588	Gor	93	05C314	$^{\sim}$ xHERMITE	461
2F2E6       GOTOLABEL       306       3B68B       xHEX       461         3B57F       xGRAD       460       054314       "xHILBERT       461         099DE       xGRAMSCHMIDT       22C006       "HILBERTNOCK       415         3C5AE       xGRAPH       460       3C9C1       xHISTOGRAM       461         2F341       GraphicExit       25636       HISTON?       461         098006       "GREDUCE       3E1CA       xHISTPLOT       461         255A1       Grey?       94       3B12C       xHMS+       461         00A0AB       xGRIDMAP       460       3B14C       xHMS+       461         38C1B       xGROB       3B10C       xHMS>       461         3317F       grob       11       39405       xHOME       461         2607B       GROB!       90       08D92       HOMEDIR       169         26080       GROB!ZERO       91       3E3006       "HORN1       169         26080       GROB!ZERODRP       91       4A8006       "HORNASIN!       169         25EDB       GROB+#       90       4A9006       "HORNASIN!       461         25EDB       GROB-JSP       9	107007	^GOSPER?	425	059314	~xHESS	461
3B57F	34A31	GOTO	129	231006	^HESSIAN	415
3B57F	2F2E6	GOTOLABEL	306	3B68B	xHEX	461
3C5AE	3B57F		460	054314	$^{\sim}$ xHILBERT	461
2F341       GraphicExit       25636       HISTON?         098006       ^GREDUCE       3E1CA       xHISTPLOT       461         255A1       Grey?       94       3B12C       xHMS+       461         00A0AB       xGRIDMAP       460       3B14C       xHMS-       461         38C1B       xGROB       3B10C       xHMS-       461         3317F       grob       11       39405       xHOME       461         2607B       GROB!       90       08D92       HOMEDIR       169         26080       GROB!ZERO       91       3E3006       ^HORN1       169         26080       GROB!ZERODRP       91       4A8006       ^HORNASIN!       169         26080       GROB!ZERODRP       91       4A8006       ^HORNASIN!       169         25EDB       GROB+#       90       4A9006       ^HORNASIN!       169         25EDB       GROB-BH#       91       4A6006       ^HORNASIN!       169         25EDB       GROB-BH#       91       4A2006       ^HORNASIN!       169         25EDB       GROBDISP       91       4A2006       ^HORNASIN!       461         07C314       *XGROBADD	0090DE	xGRAMSCHMIDT		22C006	^HILBERTNOCK	415
098006         ^GREDUCE         3E1CA         xHISTPLOT         461           255A1         Grey?         94         3B12C         xHMS+         461           00A0AB         xGRIDMAP         460         3B14C         xHMS-         461           38C1B         xGROB         3B10C         xHMS>         461           3317F         grob         11         39405         xHOME         461           2607B         GROB!         90         08D92         HOMEDIR         169           26080         GROB!ZERO         91         3E3006         ^HORN1         169           26080         GROB!ZERODRP         91         4A8006         ^HORNASIN!         169           26080         GROB!ZERODRP         91         4A8006         ^HORNASIN!         169           25EDB         GROB+#         90         4A9006         ^HORNASIN!         169           25EDB         GROB+#         91         4A6006         ^HORNASIN!         169           25EDB         GROB-H#         91         4A6006         ^HORNCOS!         160           25EDB         GROB-Menu         293,95         037314         ~HORNER         461           07C314 </td <td>3C5AE</td> <td>xGRAPH</td> <td>460</td> <td>3C9C1</td> <td>xHISTOGRAM</td> <td>461</td>	3C5AE	xGRAPH	460	3C9C1	xHISTOGRAM	461
255A1       Grey?       94       3B12C       xHMS+       461         00A0AB       xGRIDMAP       460       3B14C       xHMS>       461         38C1B       xGROB       3B10C       xHMS>       461         3317F       grob       11       39405       xHOME       461         2607B       GROB!       90       08D92       HOMEDIR       169         26080       GROB!ZERO       91       3E3006       ^hORNA       169         368E7       GROB!ZERODRP       91       4A8006       ^hORNASIN!       169         2EFDB       GROB+#       90       4A9006       ^hORNASIN!       169         2EFDB       GROB+#       91       4A6006       ^hORNASIN!       161         2E5DB       GROB-BDISP       91       4A2006       ^hORNASIN!       461         07C314       *XGROBADD       460       2C0006       ^hORNEXP!       387         08F007       *GROBADDext       92       4A1006       ^hORNEXP!       387         08F0B0       *grobCheckKey       90       4A3006       ^hORNEXP!       378         08F0B0       *GROBDIM       90       4EEF8       HSCALE <td< td=""><td>2F341</td><td>GraphicExit</td><td></td><td>25636</td><td>HISTON?</td><td></td></td<>	2F341	GraphicExit		25636	HISTON?	
00A0AB       XGRIDMAP       460       3B14C       xHMS-       461         38C1B       XGROB       3B10C       xHMS>       461         3317F       grob       11       39405       xHOME       461         2607B       GROB!       90       08D92       HOMEDIR       169         26080       GROB!ZERO       91       3E3006       ^HORN1         368E7       GROB!ZERODRP       91       4A8006       ^HORNASIN!         2EFDB       GROB+       90       4A9006       ^HORNASIN!         2F342       GROB+#       91       4A2006       ^HORNATAN!         25ED8       GROB>GDISP       91       4A2006       ^HORNATAN!         25ED8       GROB>GDISP       91       4A2006       ^HORNATAN!         25EDB       GROB>Menu       293, 95       037314       ~XHORNER       461         07C314       ~XGROBADDEXt       92       4A1006       ^HORNERlext       424         0BF007       ^GROBADDExt       92       4A1006       ^HORNEXP!       387         0870B0       ~grobCheckKey       90       4A3006       ^HORNEXP!       378         0870B0       ~grobCheckKey       90       4A3	098006	^GREDUCE		3E1CA	xHISTPLOT	461
38C1B       xGROB       3B10C       xHMS>       461         3317F       grob       11       39405       xHOME       461         2607B       GROB!       90       08D92       HOMEDIR       169         26080       GROB!ZERO       91       3E3006       ^HORN1       169         368E7       GROB!ZERODRP       91       4A8006       ^HORNASIN!       18         2EFDB       GROB+       90       4A9006       ^HORNASIN!       19         2EFDB       GROB+#       91       4A6006       ^HORNATAN!       19         25ED8       GROB>GDISP       91       4A2006       ^HORNCOS!       19         2E0D5       Grob>Menu       293,95       037314       ~XHORNER       461         07C314       ~XGROBADD       460       2C0006       ^HORNEXP!       387         0860B0       ~grobAlerticon       90       3E2006       ^HORNEXP!       387         0870B0       ~grobCheckKey       90       4A3006       ^HORNEXP!       378         0870B0       ~grobCheckKey       90       4A3006       ^HORNEXP!       422         26085       GROBDIM       90       2EEF8       HSCALE       42	255A1	Grey?	94	3B12C	xHMS+	461
3317F   grob   11   39405	00A0AB	xGRIDMAP	460	3B14C	xHMS-	461
2607B       GROB!       90       08D92       HOMEDIR       169         26080       GROB!ZERO       91       3E3006       ^HORN1	38C1B	xGROB		3B10C	xHMS>	461
2607B       GROB!       90       08D92       HOMEDIR       169         26080       GROB!ZERO       91       3E3006       ^HORN1	3317F	grob	11	39405	xHOME	461
368E7       GROB!ZERODRP       91       4A8006       ^HORNASIN!         2EFDB       GROB+       90       4A9006       ^HORNASIN!!         2F342       GROB+#       91       4A6006       ^HORNATAN!         25ED8       GROB>GDISP       91       4A2006       ^HORNCOS!         2E0D5       Grob>Menu       293,95       037314       ~xHORNER       461         07C314       ~xGROBADD       460       2C0006       ^HORNER1ext       424         0BF007       ^GROBADDext       92       4A1006       ^HORNEXP!       387         0860B0       ~grobAlerticon       90       3E2006       ^HORNEXT!       378         0870B0       ~grobCheckKey       90       4A3006       ^HORNEXIN!       378         36C68       GROBDIM       90       2EEF8       HSCALE       422         096006       ^GSOLVE       384       3FB006       ^HORNSIN!       422         2608A       GsstFIN       33175       hxs       10         2558D       Gxor       94       2F0EE       HXS#HXS       58         3C7D8       xGXOR       461       2F0EF       HXS<+HXS	2607B	GROB!	90	08D92	HOMEDIR	169
2EFDB       GROB+       90       4A9006       ^HORNASIN1!         2F342       GROB+#       91       4A6006       ^HORNATAN!         25ED8       GROB>GDISP       91       4A2006       ^HORNCOS!         2E0D5       Grob>Menu       293, 95       037314       ~xHORNER       461         07C314       ~xGROBADD       460       2C0006       ^HORNER1ext       424         0BF007       ^GROBADDext       92       4A1006       ^HORNEXP!       387         0860B0       ~grobAlertIcon       90       3E2006       ^HORNEXP!       378         0870B0       ~grobCheckKey       90       4A3006       ^HORNEXP!       378         0870B0       ~grobCheckKey       90       4A3006       ^HORNEXP!       378         0870B0       ~grobCheckKey       90       4A3006       ^HORNEXP!       378         36C68       GROBDIM       90       2EEF8       HSCALE       422         096006       ~GSOLVE       384       3FB006       ~HSCALE       422         2608A       Gsstfin       33175       hxs       10         255BD       Gxor       94       2F0EE       HXS#HXS       58         3	26080	GROB!ZERO	91	3E3006	^HORN1	
2F342       GROB+#       91       4A6006       ^HORNATAN!         25ED8       GROB>GDISP       91       4A2006       ^HORNCOS!         2E0D5       Grob>Menu       293, 95       037314       ~xHORNER       461         07C314       ~xGROBADD       460       2C0006       ^HORNER1ext       424         0BF007       ^GROBADDext       92       4A1006       ^HORNEXP!       387         0860B0       ~grobAlertIcon       90       3E2006       ^HORNEXP!       378         0870B0       ~grobCheckKey       90       4A3006       ^HORNLN!       ***         26085       GROBDIM       90       4A3006       ^HORNSIN!       ***         36C68       GROBDIMw       90       2EEF8       HSCALE       ***         096006       ^GSOLVE       384       3FB006       ^HSECO2RCext       422         2608A       GsstFIN       33175       hxs       10         2558D       Gxor       94       2F0EE       HXS#HXS       58         3C7D8       xGXOR       461       2F0EF       HXS< <hxs< td="">       59         25EDD       H/WKey&gt;KeyOb       2EFCC       HXS==HXS       58         0ED006</hxs<>	368E7	GROB!ZERODRP	91	4A8006	^HORNASIN!	
25ED8       GROB>GDISP       91       4A2006       ^HORNCOS!         2E0D5       Grob>Menu       293, 95       037314       ~xHORNER       461         07C314       ~xGROBADD       460       2C0006       ^HORNER1ext       424         0BF007       ^GROBADDext       92       4A1006       ^HORNEXP!       387         0860B0       ~grobAlertIcon       90       3E2006       ^HORNEXP!       378         0870B0       ~grobCheckKey       90       4AA006       ^HORNLN!       78         26085       GROBDIM       90       4A3006       ^HORNSIN!       78         36C68       GROBDIMW       90       2EEF8       HSCALE       78         096006       ^GSOLVE       384       3FB006       ^HSECO2RCext       422       78       78         258DD       Gxor       94       2F0EE       HXS#HXS       58       78       78         25EDC       H/W>KeyCode       205       2EFCF       HXS <hxs< td="">       59       79       78       78       78       78       78       79       78       78       78       78       78       78       78       78       78       78       78       78</hxs<>	2EFDB	GROB+	90	4A9006	^HORNASIN1!	
2E0D5       Grob>Menu       293,95       037314       ~xHORNER       461         07C314       ~xGROBADD       460       2C0006       ^hORNERlext       424         0BF007       ^GROBADDext       92       4A1006       ^hORNEXP!       387         0860B0       ~grobAlertIcon       90       3E2006       ^hORNEXP!       378         0870B0       ~grobCheckKey       90       4AA006       ^hORNLN!       378         26085       GROBDIM       90       4A3006       ^hORNSIN!       480         36C68       GROBDIMW       90       2EEF8       HSCALE       422         096006       ^GSOLVE       384       3FB006       ^hSECO2RCext       422         2608A       GSSTFIN       33175       hxs       10         2558D       Gxor       94       2F0EE       HXS#HXS       58         3C7D8       xGXOR       461       2F0EF       HXS<=HXS	2F342	GROB+#	91	4A6006	^HORNATAN!	
07C314       ~xgROBADD       460       2C0006       ^hORNER1ext       424         0BF007       ^GROBADDext       92       4A1006       ^hORNEXP!       387         0860B0       ~grobAlertIcon       90       3E2006       ^hORNEXt       378         0870B0       ~grobCheckKey       90       4AA006       ^hORNLN!       378         26085       GROBDIM       90       4A3006       ^hORNSIN!       448       448         36C68       GROBDIMW       90       2EEF8       HSCALE       422       422         2608A       GSSOLVE       384       3FB006       ^hSECO2RCext       422         2608A       GSSTFIN       33175       hxs       10         2558D       Gxor       94       2F0EE       HXS#HXS       58         3C7D8       xGXOR       461       2F0EF       HXS<=HXS	25ED8	GROB>GDISP	91	4A2006	^HORNCOS!	
0BF007         ^GROBADDext         92         4A1006         ^HORNEXP!         387           0860B0         ~grobAlertIcon         90         3E2006         ^HORNEXT         378           0870B0         ~grobCheckKey         90         4AA006         ^HORNLN!         ***           26085         GROBDIM         90         4A3006         ^HORNSIN!         ***           36C68         GROBDIMW         90         2EEF8         HSCALE         ***           096006         ^GSOLVE         384         3FB006         ^HSECO2RCext         422           2608A         GSSTFIN         33175         hxs         10           2558D         Gxor         94         2F0EE         HXS#HXS         58           3C7D8         xGXOR         461         2F0EF         HXS<=HXS	2E0D5	Grob>Menu	293, 95	037314	~xHORNER	461
0860B0       ~grobAlertIcon       90       3E2006       ^HORNext       378         0870B0       ~grobCheckKey       90       4AA006       ^HORNLN!         26085       GROBDIM       90       4A3006       ^HORNSIN!         36C68       GROBDIMW       90       2EEF8       HSCALE         096006       ^GSOLVE       384       3FB006       ^HSEC02RCext       422         2608A       GSStFIN       33175       hxs       10         2558D       Gxor       94       2F0EE       HXS#HXS       58         3C7D8       xGXOR       461       2F0EF       HXS<=HXS	07C314	~xGROBADD	460	2C0006	^HORNER1ext	424
0870B0       ~grobCheckKey       90       4AA006       ^hORNLN!         26085       GROBDIM       90       4A3006       ^hORNSIN!         36C68       GROBDIMW       90       2EEF8       HSCALE         096006       ^GSOLVE       384       3FB006       ^hSECO2RCext       422         2608A       GSSTFIN       33175       hxs       10         2558D       Gxor       94       2F0EE       HXS#HXS       58         3C7D8       xGXOR       461       2F0EF       HXS<=HXS	0BF007	^GROBADDext	92	4A1006	^HORNEXP!	387
26085       GROBDIM       90       4A3006       ^HORNSIN!         36C68       GROBDIMW       90       2EEF8       HSCALE         096006       ^GSOLVE       384       3FB006       ^HSECO2RCext       422         2608A       GSSTFIN       33175       hxs       10         2558D       Gxor       94       2F0EE       HXS#HXS       58         3C7D8       xGXOR       461       2F0EF       HXS<=HXS	0860B0	$\sim$ grob $A$ lert $I$ con	90	3E2006	^HORNext	378
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	0870B0	~grobCheckKey	90	4AA006	^HORNLN!	
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	26085	GROBDIM	90	4A3006	^HORNSIN!	
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	36C68	GROBDIMw	90	2EEF8	HSCALE	
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	096006	^GSOLVE	384	3FB006	^HSECO2RCext	422
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	2608A	GsstFIN		33175	hxs	10
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	2558D	Gxor	94	2F0EE	HXS#HXS	58
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	3C7D8	xGXOR	461		HXS<=HXS	59
0ED006 ^H>Z 328 05A03 HXS># 22 001100 xH→ 478 2EFC0 HXS>\$ 46	25EDC	H/W>KeyCode	205	2EFCF	HXS <hxs< td=""><td>59</td></hxs<>	59
001100 xH $\rightarrow$ 478 2EFC0 HXS>\$ 46	25EDD	H/WKey>KeyOb		2EFCC	HXS==HXS	58
001100 xH $\rightarrow$ 478 2EFC0 HXS>\$ 46	0ED006	^H>Z	328	05A03	HXS>#	22
005100 xH $\rightarrow$ A 478 2EFC1 hxs $>$ \$	001100	$xH{\longrightarrow}$	478	2EFC0	HXS>\$	46
	005100	$xH \rightarrow A$	478	2EFC1	hxs>\$	46

Addr.	Name	Page	Addr.	Name 1	Page
2EFCA	HXS>%	31	03D004	^IfEnterKeyPress	260
2EFCE	HXS>=HXS	59	387AC	xIFERR	462
2EFCD	HXS>HXS	59	01B0AB	xIFFT	462
336C5	HXSREAL	16	027004	^IfGetCurrentFieldValue	258
417006	^HYP2EXPext	356	02C004	^IfGetFieldChooseData	258
02B0DE	xHYPERBOLIC		02D004	^IfGetFieldChooseDecomp	259
104007	^HYPERGEO	393	02B004	^IfGetFieldDecompObject	
39B3B	xi	462	030004	^IfGetFieldInternalVa	
3F0B7	xI>R	462	028004	^IfGetFieldMessageHan	258
20C006	^IABCUV	414	02A004	^IfGetFieldObjectsType	258
031314	~xIABCUV	461	045004	^IfGetFieldPos	261
0120DE	xIBASIS		02E004	^IfGetFieldResetValue	259
10D007	^IBERNOULLI	394	029004	^IfGetFieldType	258
0060DE	xIBERNOULLI	461	026004	^IfGetFieldValue	258
00B314	~xIBP	461	032004	^IfGetNbFields	259
2C5006	^IBP	390	04C004	^IfGetPrlgFromTypes	259
03B314	~xICHINREM	461	04A004	^IfInitDepth	261
217006	^ICHINREM	414	03A004	^IfKeyCalc	260
2F0EC	ICMPDRPRTDRP	71	037004	^IfKeyChoose	259
33143	id	10	038004	^IfKeyEdit	260
05BE9	ID>\$	46	03B004	^IfKeyInvertCheck	260
2E11B	Id>Menu	293, 95	039004	^IfKeyTypes	260
05F2E	ID>TAG	61	020004	^IfMain	257
272F3	ID_EQ	116	042004	^IfMain2	261
27937	ID_SIGMADAT	116	0050B0	~IFMenuRow1	257
3E9006	^IDIV2		0060B0	~IFMenuRow2	257
027314	~xIDIV2	461	03C004	^IfONKeyPress	260
33887	IDLISTOB	19	043004	^IfPutFieldsOnStack	261
3C02E	XIDN	462	035004	^IfReset	259
33143	idnt	10	048004	${}^{ ilde{ ii}}}}}}}}}}}}}}}}}}}}}}}}}}} {}}}}}}}}}$	261
2B0CC	idntcase	145	047004	^IfSetAllLabelsMessages	
45C006	^IDNTEXEC	423	025004	^IfSetCurrentFieldValue	258
191006	^IDNTLAM?	201	036004	^IfSetField	259
2B0EF	idntlamcase	145	044004	^IfSetFieldPos	261
334D1	IDREAL	15	02F004	^IfSetFieldResetValue	259
3387D	IDREALOB	18	024004	^IfSetFieldValue	258
0716B	IDUP	128, 150	021004	^IfSetFieldVisible	258
20A006	^IEGCD		023004	^IfSetGrob	258
02F314	~xIEGCD	462	03F004	^IfSetHelpString	260
3D6006	^IEGCDext	330	022004	^IfSetSelected	258
37F48	xIF	462	040004	^IfSetTitle	260
034004	^IfCheckFieldtype	259	041004	^IfSetTitle2	
033004	^IfCheckSetValue	259	396A4	xIFT	462
049004	^IfCreateTitleGrob		395F3	XIFTE	462
031004	^IfDisplayFromData	259	04B004	^IfTet	
046004	^IfDisplayFromData2		011314	~xILAP	462
0BC002	^IFEDispField		08A007	^ILAPDELTA	
3807D	XIFEND		08B007	^ILAPEXP	392

Addr.	Name	Page	Addr.	Name	Page
08C007	^ILAPEXPSC		2EF97	InsertEcho	302
088007	^ILAPext	392	363006	<pre>^insertrow[]</pre>	343
089007	^ILAPRAText		362006	^INSERTROW[]	343
2F0D4	ILnot?	199	2FC006	^INSERT{}N	72
3B87E	XIM	462	3F007	$\times INT$	462
0100DE	xIMAGE		2D3006	^INT3	390
503006	^xIMext	348	0290DE	XINTEGER	
25EDF	ImmedEntry?	278	33805	INTEGER337	17
35BAF	INCOMPDROP	71	3D47E	imesINTEGRAL	
3E54C	xINCR	462	3DD006	^INTEGRext	391
2F343	IncrLAMPKNO		06B4E	INTEMNOTREF?	171
3C33E	XINDEP	462	01E0E8	~INTEMPOB?	171
2F0E8	INDEPVAR	317	516006	^INTERNALARG2	
07221	INDEX@	151	582006	^INTERNALERR	403
367D9	INDEX@#-	152	2D2006	^INText	390
07270	INDEXSTO	152	586006	^INTVARERR	404
3D9006	^INEGCD	330	004314	~xINTVX	462
2E2006	^INFINIext	419	3A32B	XINV	462
04C0AB	xINFORM	462	402006	^INVAL2	423
394C8	INHARDROM?	179	583006	^INVALIDOP	403
2EEE6	InitEd&Modes	314	52A006	^xINVext	348
2EEE7	InitEdLine	303, 314	2609E	INVGROB	91
2EEE8	InitEdModes	314	2E25C	InvLabelGrob	90
092DB	InitEnab		24B006	^INVMOD	
2F0E7	InitIOEnv		074314	~xINVMOD	462
25EE0	InitMenu	291	350006	^INXREDext	341
25EE1	InitMenu%	291	00110	IOC	
26256	INITMKFONT		2F346	IOCheckReal	
2B709	InitPOLVars		81006	IOCsave	
2F075	InitSysUI		3AC3D	xIP	462
25EE2	<pre>InitTrack:</pre>	289	2EF006	^ipi	420
26251	${ t Init Virtual Stack}$		029314	$^{\sim}$ xIQUOT	462
366E9	INNER#1=	71	0011F	IRAM@	
054AF	INNERCOMP	71	800F5	IRAMBUFF	
3BADA	INNERCOMP>%	71	0011A	IRC	
35C68	INNERDUP	71	02B314	$^{\sim}$ xIREMAINDER	462
3D3006	^INPARTFRAC	391	113007	^IROOTS	382
3EEBD	XINPUT	462	2D8006	^IRRQ#ULTIMATE	422
2F154	input\$	212	160006	^IRXC2	
2F344	InputLAttn		15F006	^IRXCext	37
2F345	InputLEnter		2C7006	^IS_SQRT?	424
2EF5F	InputLine	212	2C9006	^IS_XROOT?	424
2F155	input{}	212	0C8007	^ISIDREAL?	408
25790	INSERT?	302	2CC006	^ISNT_IDNT?	
366006	^INSERT[]COL[]	344	3E648	xISOL	462
365006	^INSERT[]ROW[]	344	4C5006	^ISOL1	396
25795	INSERT_MODE	302	4C7006	^ISOL2ext	396
364006	^INSERTCOL[]	344	4C6006	^ISOLALL	396

Addr.	Name	Page	Addr.	Name	Page
584006	^ISOLERR	404	3ECE4	xKGET	463
00D0DE	xISOM		394F1	xKILL	463
4E0006	^ISPOLYNOMIAL?	397	260A3	KILLGDISP	273
218006	^ISPRIME	411	2F34D	KINVISLF	181
03C314	~xISPRIME?	462	2F34E	KVIS	181
283FC	ISTOP-INDEX	152	2F34F	KVISLF	181
07249	ISTOP@	152	361006	^la+ELEMsym	343
07295	ISTOPSTO	152	3C5C9	xLABEL	463
04D004	^IsUncompressDataString	261	25877	LabelDef!	290
0E9006	^IsV>V?	399	06C003	^laDELROW	
34A22	IT	138	06E003	^laGPROW	
34B3E	ITE	139	05D314	~xLAGRANGE	463
34ABE	ITE_DROP	139	235006	^LAGRANGE	415
36DDF	j%0=case	142	3E6006	^LAGRANGEext	379
36D3A	jEQcase	144	06D003	^laINSROW	
07258	JINDEX@	152	3314D	lam	10
072AD	JINDEXSTO	152	2F205	laMGET0	
050314	~xJORDAN	463	27142	LAMLNAME	
380006	^JORDAN	345	25F95	LANGUAGE>	179
04D87	JstGetTHEMESG	157	0010DD	imesLANGUAGE $ ightarrow$	463
04D87	JstGETTHEMSG	157	010314	~xLAP	463
07264	JSTOP@	152	087007	^LAPext	392
072C2	JSTOPSTO	152	058314	~xLAPL	463
2F347	JUMPBOT	280	230006	^LAPLACIAN	415
2F348	JUMPLEFT	280	397E5	xLAST	463
2F349	JUMPRIGHT	280	362B6	LAST\$	48
2F34A	JUMPTOP	280	80F5A	LASTARGCOUNT	
2F34B	KDispRow2		2BC006	^LASTCOMP	68
2F34C	KDispStatus2		25908	LastMenuDef!	289
25EE4	KeepUnit	82	2590D	LastMenuDef@	289
00F0DE	xKER	-	260A8	LastMenuRow!	288
2F0E6	KERMOPEN		260AD	LastMenuRow@	289
00C10	kermpktmsg		25EE7	LastNonNull	168
00C0E	kermrecvmsq		2F351	LASTPT?	100
00C0D	kermsendmsg		08326	LASTRAM-WORD	168
3EE2C	xKERRM	463	0670AB	xlbmol	
39854	xKEY	463	255F6	LBoxB	94
25EE5	Key>StdKey0b	208	255EC	LBoxG1	94
25EE6	Key>U/SKeyOb	208	255F1	LBoxG2	94
255006	^KEYEVAL	208	255E7	LBoxW	94
07B314	~xKEYEVAL	463	255FB	LBoxXor	94
25EE3	KEYINBUFFER?	206	33A5D	lbrac	01
04E004	^KeyLookup	200	3C866	xLCD>	463
25949	KeyOb!		02D314	~xLCM	463
2593F	KeyOb:		375006	^LCPROG2M	337
2594E	KeyOb@		055314	~xLCXM	463
06D0AB	xKEYTIME→	463	017100	xLC^C	479
25F2A	Keyword?	400	017100	~xLDEC	463
ZJFZA	relance:		012314	AUDEC	400

Addr.	Name	Page	Addr.	Name	Page
081007	^LDECSOLV	391	4AF006	^LIMDL!	
082007	^LDEGENE	391	4B0006	^LIMDLINF!	
083007	^LDEPART	391	481006	^LIMEQU!	386
084007	^LDSSOLVext	391	482006	^LIMEQU0!	
0B1002	^LEDispItem	245	480006	^LIMEQUFR!	386
0B2002	^LEDispList	245	475006	^LIMERRO!	
0B3002	^LEDispPrompt	244	476006	^LIMERR1!	
25FF9	LEFT\$3x5	97	484006	^LIMERR10!	
26008	LEFT\$3x5Arrow	97	4BC006	^LIMERR6!	
2601C	LEFT\$3x5CR	97	496006	^LIMEXP!	
26012	LEFT\$3x5CRArrow	97	49B006	^LIMFLOOR!	
25FFE	LEFT\$5x7	97	4AD006	^LIMHORN!	
26003	LEFT\$5x7Arrow	97	4B1006	^LIMINFSIGN!	
26017	LEFT\$5x7CR	97	487006	^LIMINV!	
2600D	LEFT\$5x7CRArrow	97	494006	^LIMINVLN!	
2F352	LEFTCOL	279	005314	~xLIMIT	463
234006	^LEGENDRE	415	477006	^LIMIT!	386
05A314	$^{\sim}$ xLEGENDRE	463	46D006	^LIMIText	
05636	LEN\$	47	474006	^LIMITNOVX!	
0567B	LENCOMP	68	473006	^LIMITX!	
05616	LENHXS	57	47C006	^LIMLIM!	386
17B006	^LENMATRIX	343	47E006	^LIMLIM1!	
2D9006	^LESSCOMPLEX?	424	495006	^LIMLN!	
0B6006	^LFCProd	375	4B2006	^LIMMAX!	
032314	~xLGCD	463	485006	^LIMNEG!	
0160AB	xLIBEVAL	463	489006	^LIMPOW!	
3EB42	xLIBS	463	48F006	^LIMPROF!	
4DE006	^LIDNText	397	491006	^LIMPROF0!	
4EC006	^LIDNTLVAR	398	492006	^LIMPROF1!	
256006	^LIFCext	395	493006	^LIMPROF2!	
2F21C	Lift		48E006	^LIMPROFEND!	387
0BC006	^LiftGeneral	376	486006	^LIMRAC!	
0BB006	^LiftLinear	376	4A4006	^LIMSC0!	
0B5006	^LiftZAdic	375	4A5006	^LIMSC1!	
49E006	^LIM#VARX!	387	472006	^LIMSERIES!	386
490006	^LIM%#!	387	497006	^LIMSINCOS!	
48B006	^LIM*!		4B5006	^LIMSORT!	
483006	^LIM+-!	387	48A006	^LIMSQ!	
488006	^LIM/!		49A006	^LIMSQRT!	
49C006	^LIMABS!		478006	^LIMSTEP1!	386
4A0006	^LIMALPHA!		479006	^LIMSTEP2!	
499006	^LIMASIN!		47A006	^LIMSTEP3!	
498006	^LIMATAN!		47B006	^LIMSTEP4!	
4A7006	^LIMATAS!		4BE006	^LIMVAL!	387
49F006	^LIMBETA!		4BD006	^LIMVALOBJ!	387
47F006	^LIMCMPL!	386	4C3006	^LIMVAR!	387
4B3006	^LIMCOMP!		014314	~xLIN	463
48C006	^LIMDIVPC!	387	3C68C	xLINE	463

Addr.	Name	Page	Addr.	Name	Page
101007	^LINEARAPPLY	425	319006	^LNEXPA/	369
102007	^linearapply		31A006	^LNEXPA^	369
2556A	LineB	93	522006	^xLNext	349
2F353	LINECHANGE		4AB006	^LNOBJ!	
2556F	LineG1	93	3AB2F	xLNP1	464
25574	LineG2	93	41A006	^LNP12LN	351
2EFA0	LINEOFF	92	25EE8	LoadTouchTbl	292
2EFA3	LINEOFF3	92	25EE9	LockAlpha	278
2EF9F	LINEON	92	3AA73	xLOG	464
2EFA2	LINEON3	92	41B006	^LOG2LN	351
25565	LineW	93	3E239	xLOGFIT	464
25579	LineXor	93	377C5	Lookup	70
31B006	^LINEXPA	355	377DE	Lookup.1	70
300006	^LINEXPext	355	07334	LOOP	151
3E214	xLINFIT	463	05149	Loop	
0150AB	xLININ	463	269006	^LOP1ext	421
080007	^LINSOLV	342	26A006	^LOPAext	421
052314	~xLINSOLVE	463	3F2006	^LOPDext	420
33139	list	10	10F006	^LOPMext	421
2F354	List		01118	LowBat?	178
2DC006	^LIST10-10		2B1006	^LPGCDext	358
2DB006	^LIST1i-1-i		49D006	^LPROF!	
17A006	^LIST2MATRIX	338	0320AB	xLQ	464
3BAC1	xLIST>	463	3DF83	xLR	464
3343B	LISTCMP	14	4AE006	^LRDM!	
45F006	^LISTEXEC	421	3DE006	^LRDMext	378
460006	^LISTEXEC1	421	015100	xLR~R	479
2DA006	^LISTIRRQ	422	02B0AB	xLSQ	464
3346D	LISTLAM	14	0300AB	xLU	464
104006	^LISTMAXext	399	06A314	~xLVAR	464
4EE006	^LISTOPext	398	4F0006	^LVARDext	398
4ED006	^LISTOPRAC	398	4E7006	^LVARext	397
4EF006	^LISTOPSQRT	398	4C0006	^LVARXNX2!	387
0FD006	^ListPos	69	4E3006	^LVARXNX2ext	
2F24E	LISTRCL	166	4DF006	^LVARXNXext	397
33431	LISTREAL	14	386006	^m-1&m+1	359
26B006	^LISTSECOext	421	2AE32	M-1stcasechs	143
001007	^ListToArry	65	35AE2	MACRODCMP	
4E5006	^LLVARDext	397	051314	~xMAD	464
3AA01	xLN	464	321006	^MADD	339
45A006	^LN2ATAN	352	242006	^MADDTMOD	416
456006	^LN2ext	357	37E006	^MADJ	345
06D314	~xLNAME	464	229006	^MADNOCK	415
527006	^LNATANext	349	07F314	~xMAIN	
31D006	^LNCOLCext	355	377006	^make2dmatrix	337
016314	$\sim$ xLNCOLLECT	464	376006	^MAKE2DMATRIX	337
316006	^LNEXPA	355	373006	^MAKEARRY	337
318006	^LNEXPA*	369	2E189	MakeBoxLabel	95

Addr.	Name	Page	Addr.	Name	Page
2E1EB	MakeDirLabel	95	34C006	^MATREF	341
260B2	MAKEGROB	91	34F006	^MATREFRREF	341
2E24D	MakeInvLabel	95	340006	^MATREPL	343
2E2AA	MakeLabel	293, 96	36D006	^MATRIX-COL	344
2F0DB	MAKEPICT#	91	36B006	^MATRIX-ROW	344
2AB006	^MAKEPROFOND	378	178006	^MATRIX2ARRAY	65
2F355	MAKEPVARS	316	179006	^MATRIX2LIST	338
2E166	MakeStdLabel	95	35F006	^MATRIX>DIAG	343
01C100	xMAKESTR	479	369006	^MATRIXCSWAP	344
357006	^MAKESYSText	342	360006	^MATRIXDIAG>	343
3B02E	xMANT	464	174006	^MATRIXDIM	341
066314	~xMAP	464	367006	^MATRIXRCI	341
27D5D	MARKGROB	90	368006	^MATRIXRCIJ	341
326006	^MAT*	339	1E7006	^MATRIXRISCH	412
248006	^MAT*MATMOD		36A006	^MATRIXRSWAP	344
32A006	^MAT*SCL	339	349006	^MATRNORM	340
246006	^MAT*SCMOD		34B006	^MATRREF	341
320006	^MAT+	339	335006	^MATSQUARE	340
322006	^MAT-	339	33E006	^MATSUB	343
333006	^MAT/	340	339006	^MATTRACE	340
332006	^MAT/SCL	340	33C006	^mattran	340
32C006	^MAT^	340	33B006	^MATTRAN	65
2639B	MATATLOOP		33D006	^mattrn	340
3DB04	xMATCHDN	475	33A006	^MATTRN	340
376B7	matchob?	69	3ADA5	xMAX	464
376C1	matchob?Lp		39AE4	xMAXR	464
334006	^MATCHS	340	3DEE1	xMAXSIGMA	464
3DAD0	xMATCHUP	464	0760AB	xMCALC	464
34A006	^MATCNORM	340	16D006	^MDIMS	64
372006	^MATCON	337	35FD8	MDIMSDROP	64
336006	^MATCONJ	340	3DEFC	xMEAN	464
32D006	^MATCROSS	340	05F61	MEM	178
346006	^MATDET	340	3E8C1	xMEM	464
32E006	^MATDOT	340	33111	MEMERR	10
37F006	^MATEGV	345	2EE8B	MENoP&FixDA1	
37C006	^MATEGVL	345	2EF59	MENP&FixDA12	
17D006	^MATEXPLODE	339	3E9D4	xMENU	464
348006	^MATFNORM	340	0В9007	^MENUARIT1	409
02F0DE	xMATHS		0B5007	^MENUBASE1	409
371006	^MATIDN	337	0B3007	^MENUCHOOSE	409
338006	^MATIM	340	0B2007	^MENUCHOOSE?	409
34E006	^MATINV	340	0B6007	^MENUCMPLX1	409
083314	~xMATR		25845	MenuDef@	289
34D006	^MATRANK	341	0BC007	^MENUDIFF1	410
344006	^MATRANM	337	0BB007	^MENUEXPLN1	410
347006	^MATRDET	340	08D007	^MENUext	409
337006	^MATRE	340	0В4007	^MENUGENE1	409
341006	^MATREDIM	337	275FD	MenuKey	293

Addr.	Name	Page	Addr.	Name	Page
2589F	MenuKeyLS!	290	1C9006	^METADERACH	389
2588B	MenuKeyNS!	290	1C6006	^METADERACOS	389
25890	MenuKeyNS@	290	1B8006	^METADERALOG	389
258B3	MenuKeyRS!	290	1C8006	^METADERASH	389
27620	MenuMaker	291	1C5006	^METADERASIN	389
0B8007	^MENUMAT1	409	1C7006	^METADERATAN	389
2EEFC	MENUOFF	273	1CA006	^METADERATH	390
2EEFD	MENUOFF?	273	1C0006	^METADERCOS	389
260B7	MenuRow!	288	1C3006	^METADERCOSH	389
260BC	MenuRow@	288	1AF006	^METADERDER	388
25863	MenuRowAct!	289	1B4006	^METADEREXP	388
0BA007	^MENUSOLVE1	410	1AE006	^METADERFCN	388
0B7007	^MENUTRIG1	409	1B1006	^METADERI3	388
07A314	~xMENUXY	464	1B0006	^METADERI4	388
1D1006	^MENUXYext	409	1B2006	^METADERIFTE	388
2AFFB	MEQ1stcase	143	1BA006	^METADERINV	389
2B01B	MEQopscase	143	1A6006	^METADERIV	388
3EB16	xMERGE		1B5006	^METADERLN	388
0120E4	~MESRclEqn		1B6006	^METADERLNP1	388
390006	^meta-1	360	1B7006	^METADERLOG	389
2F1006	^meta-pi	420	1BB006	^METADERNEG	389
2F4006	^meta-pi/2	420	1A8006	^METADEROP	
2F5006	^meta-pi/4	420	1BF006	^METADERSIN	389
389006	^meta/2	359	1C2006	^METADERSINH	389
38D006	^meta1-sq	360	1BE006	^METADERSQ	389
387006	^meta1/meta	359	1BC006	^METADERSQRT	389
38C006	^meta2*	359	1C1006	^METADERTAN	389
397006	^meta^	361	1C4006	^METADERTANH	389
103007	^meta_cst?	393	396006	^metadiv	361
2FA006	^meta_e	420	3B1006	^MetaDiv	361
3AB006	^MetaAdd	360	39F006	^metaEQUAL?	78
393006	^metaadd	360	310006	^METAEXPEXPA	369
3BA006	^metackneg	362	39A006	^metafraction?	370
277006	^METACOMP0		2ED006	^metai	420
278006	^METACOMP1	381	459006	^metai*	360
3BD006	^metaCOMPARE	370	3A5006	^metainftype	368
276006	^METACOMPRIM	381	198006	^METAINT?	78,334
30A006	^METACOSEXPA	369	2F0006	^metaipi	420
287006	^METADEG1	382	4D8006	^METALISTVXXL	325
286006	^METADEG2	382	317006	^METALNEXPA	369
2A6006	^METADENOLCM	358	36C006	^METAMAT-ROW	344
1BD006	^METADER&NEG	388	36E006	^METAMATCSWAP	344
1AB006	^METADER*	388	351006	^METAMATRED	341
1A9006	^METADER+	388	36F006	^METAMATRSWAP	344
1AA006	^METADER-	388	274006	^METAMM2	381
1AC006	^METADER/	388	3AF006	^MetaMul	361
1AD006	^METADER^	388	273006	^METAMULMULT	
1B9006	^METADERABS	389	395006	^metamult	361

Addr.	Name	Page	Addr.	Name	Page
3B9006	^metaneg	362	00E0DE	xMKISOM	
3B7006	^MetaNeg	362	0A5006	^MKPOLY	378
2EB006	^metapi	420	4D2006	^MLISTSQFF	397
2F8006	^metapi*2	420	327006	^MMMULT	339
2F2006	^metapi/2	420	3AFCB	xMOD	465
2F3006	^metapi/4	420	3D0006	^Mod	329
3BC006	^metapi?	370	3C5006	^ModAdd	372
352006	^METAPIVOT	341	3C8006	^ModDiv	373
199006	^METAPOSINT?	78,334	3C9006	^ModDiv2	373
3B5006	^MetaPow	362	3C4006	^ModExpa	372
399006	^metapow	361	3D1006	^ModFctr	
3C3006	^metareal?	370	3CB006	^ModGcd	373
36FA6	metaROTDUP	76	275EE	Modifier	291
3BB006	^metasimp	368	25EEA	ModifierKey?	205
304006	^METASINEXPA	368	3CA006	^ModInv	373
291006	^METASOLV	383	3CC006	^ModLGCD	
293006	^METASOLV2	383	3CD006	^ModLOPD	
294006	^METASOLV4	383	3C7006	^ModMul	373
292006	^METASOLVOUT		0DB006	^ModPow	
385006	^metasplit	78	079314	~xMODSTO	465
38E006	^metasq+1	360	3C6006	^ModSub	373
38F006	^metasq-1	360	02C0DE	xMODULAR	
4D3006	^METASQFFext	397	3CF006	^MODULOMAText	
394006	^metasub	360	3CE006	^MODULOMODext	
3AD006	^MetaSub	361	0В9006	^MonicLf	375
2F356	metatail	78	2C388	MOVEVAR	
31E006	^METATANEXPA	369	121006	^MP0	423
3A2006	^metaundef	367	122006	^MPEXEC	
403006	^METAVAL2	423	120006	^MPext	372
39B006	^metaxroot	362	2DE26	mpop1%	
465006	^MEVALext	353	0770AB	xMROOT	465
253006	^MFACTORMOD	395	4D1006	^MSECOSQFF	425
0A9006	^MFactTriv	374	04E0AB	xMSGBOX	465
4CA006	^MHORNER	396	0040B1	~MsgBoxMenu	283
4CB006	^MHORNER1	396	0200DE	xMSLV	
3E4006	^MHORNext	378	0720AB	xMSOLVR	465
2B083	Mid1stcase	143	4CD006	^MSQFF	396
3AE2B	xMIN	465	323006	^MSUB	339
2625B	MINIFONT>	283	244006	^MSUBTMOD	416
0120DD	$\mathtt{xMINIFONT} {\rightarrow}$	465	31C006	^MTRIG2SYMB	355
0730AB	xMINIT	465	272006	^MULMULText	381
39B01	xMINR	465	245006	^MULTMOD	416
3DF17	xMINSIGMA	464	070314	$\sim$ xMULTMOD	465
3A9006	^minusinf	368	0750AB	xMUSER	465
2E3006	^MINUSINFext	419	328006	^MVMULT	339
3399F	MINUSONE	21	08309	MYRAMROMPAIR	
24C006	^MINVMOD		4CF006	^MZSQFF	396
0740AB	xMITM	465	4D0006	^MZSQFF1	396

Addr. Name Page Addr. Nam	e Page
35FB0 N+1DROP 75, 106 33161 NINE	10
36B67 NcaseSIZEERR 146 331C5 NINE	TEEN 11
36BAA NcaseTYPEERR 146 3F264 xNIP	465
2C2CB nCOLCTQUOTE 2EF07 nmet	asyms 199
2C2C0 nCustomMenu 293 ODEOBO ~nNul	lBind 117
0D9007	FFCMD 312
26260 nDISPSTACK 275 25EED NOAT	tn?Semi 207
01COAB xNDIST 465 256BE NOBL	INK 179
0326E NDROP 75, 106 25EEE NoEd	it?case 146
	itLine? 300
3AA006 ^NDROPminusinf 368 38D2F xNOEV.	AL>
3A8006 ^NDROPplusinf 368 25EEF NoEx	itAction 291
3B3006 ^NDROPZ0 327 27E72 noha	lt $224$
3B4006 ^NDROPZ1 327 260C1 NOHA	LTERR 157
391006 ^NDROPZERO 76 257E2 No.1q:	noreAlm
031D9 NDUP 106 119007 ^NONA	LGERR 404
3F2B5 xNDUPN 465 585006 ^NONI	NTERR 404
28143 NDUPN 106 2ADBD nono	pcase 145
	OLYSYST 404
166006 ^NDXQext 421 100007 ^NONR.	ATSUM 393
	ECMODE 407
39976 xNEG 465 581006 ^NONU	NARYERR 403
39CD5 xNEGNEG 25621 NonU	srKeyOK? 208
2F357 newBASE 06E8E NOP	128
	EPBYSTEP 407
2F0D5 NEWINDEP 3CB13 xNOT	465
4AC006 ^NEWLIMHORN 03AF2 NOT	136
33B39 NEWLINE\$ 43 34BD8 NOT?	DROP 138
361DA NEWLINE \$&\$ 47 34A59 NOT?	
361DA NEWLINE&\$ 47 34A92 NOT?	
·	SWAPDROP 138
091007 ^NEWMODULO 405 0712A NOT_	
_	UNTIL 151
	WHILE 151
090007 ^NEWVX 405 35C7C NOTA	
3831C XNEXT 465 34A13 NOTC	
	ase2drop 139
	ase2DROP 140
	asedrop 139
	aseDROP 139
	aseFALSE 140
_	aseTRUE 140
	sdrpfls 139
	ISTcase 145
	OMPcase 145
	ECOcase 145
OCB006 ^NFactorSpc 330 3F0FC xNOVA	
29E29 ngsizecase 469006 ^NR_R	

Addr.	Name	Page	Addr.	Name	Page
2BB53	nscknum2	87	05944	OCRC	179
2F38E	xnsgeneral	77	2F257	OCRC%	179
3DE09	xNSIGMA	465	3B6A6	xOCT	465
0560AB	xNSUB	465	07F007	^ODE_INT	390
35D58	NTHCOMDDUP	69	086007	^ODE_SEPAR	392
35BC3	NTHCOMPDROP	69	085007	^ODETYPESTO	391
056B6	NTHELCOMP	68	3950C	XOFF	465
37784	NTHOF	69	076AE	OFFSRRP	100
055DF	NULL\$	43	3D0BC	xOLDPRT	
0556F	NULL\$?	55	33111	ONE	10
35D94	NULL\$SWAP	46	36739	ONE#>	25
25EEC	NULL\$TEMP	46	0A6006	^ONE>POLY	378
055FD	NULL::	73	073CE	ONE_DO	151
055B7	NULLCOMP?	68	363CE	ONE_EQ	25
27AA3	NULLGROB	90	36A63	ONECOLA	131
055D5	NULLHXS	57	36B12	ONEDUP	21
272FE	NULLID	116	3657C	ONEFALSE	22
2B3AB	NULLLAM	116	369E6	ONEFALSE'	130
27FED	NullMenuKey	291	334EF	ONEHUNDRED	15
359006	^NULLVECTOR?	339	36B12	ONEONE	21
055E9	NULL{}	72	35EA2	ONESWAP	22
29E67	nultrior	50	29D18	ONE { } N	72
3BBF9	xNUM	465	09B006	^ONE { } POLY	377
2B25C	num-1=case	143	2F19B	OngoingText?	
2B11C	num0=case	142	2F2FF	OpenIO	
2B176	num1=case	143	3EC75	XOPENIO	466
2B1DF	num2=case	143	2F35D	OpenIOPrt	
2ADE0	numb1stcase	145	2F313	OpenUart?Clr	
0A0007	^NUMMODE	406	2F312	OpenUartClr	
2F35B	NUMSOLVE		03B75	OR	136
0060AB	xNUMX	465	3CA8D	xOR	466
0070AB	xNUMY	465	25EF0	OR\$	50
2C044	nWHEREDER		359E3	ORcase	139
2C039	nWHEREIFTE		3E8F0	XORDER	466
2C04F	nWHEREINTG		280C1	ORDERXY#	92
2C05A	nWHERESUM		280F8	ORDERXY%	92
2C065	nWHEREWHERE		365F9	ORNOT	136
47D006	^n{}N		05902	OSIZE	178
39CB3	ob&\$	53	0020F	OUTCINRTN	
2F35C	OB>BAKcode		3DC8C	XOVER	466
37073	0b>Seco	73	032C2	OVER	109
2F1AE	ObEdit	311	36775	OVER#+	24
197006	^OBJ2REAL	31	367C5	OVER#-	24
3BE38	xOBJ>	465	36694	OVER#0=	25
2B8BE	OBJ>R	129	36147	OVER#2+UNROL	77
374006	^OBJDIMS2MAT	337	366A8	OVER#<	25
19A006	^OBJINT?	334	35268	OVER#=	25
19B006	^OBJPOSINT?	334	348E2	OVER#=case	141

Addr.	Name	Page	Addr.	Name	Page
36725	OVER#>	26	2C1006	^PEval	424
369AA	OVER'	130	0EA006	^PEvalFast?	380
36CF4	OVER5PICK	109	0E1006	^PEvalMod	395
36183	OVERARSIZE	64	0DA007	^PEvalN/D	394
35CF4	OVERDUP	109	0AC006	^PFactor	374
36482	OVERINDEX@	152	0C5006	^PFactPowCnt	377
05622	OVERLEN\$	47	0C3006	^PFactTriv	376
35D6C	OVERSWAP	109, 109	268006	^PFEXECext	421
35D6C	OVERUNROT	109, 109	3D5006	^PFext	
351FA	OverWrF/TLp		3EAA7	xPGDIR	466
01F0DE	xP2C		0AE006	^PHFctr	374
0D8007	^P2P#	374	0B0006	^PHFctr0	374
215006	^PA2B2	330	0AF006	^PHFctr1	374
039314	~xPA2B2	466	2EA006	^pi	420
2EF62	palparse	50	39AC7	xPI	466
35B46	PALPTRDCMP		30017	PI/180	30
35B82	palrompdcmp	55	438006	^pi/2-acos	366
3C98B	xPARAMETRIC	466	43B006	^pi/2-asin	366
3EDEC	xPARITY	466	439006	^pi/2-meta	366
0CD007	^PARITYTEST		3DCFD	xPICK	466
2B475	ParOuterLoop	223	032E2	PICK	109
2EF6A	Parse.1		3F27F	xPICK3	466
2EF6B	Parse.2		3C72A	xPICT	466
2EF6E	ParseFail	51	2F258	PICTRCL	316
2EF70	ParseFail2		28A006	^PIext	73
0090AB	xPARSURFACE	466	2F6006	^pifois2	420
3D2006	^PARTFRAC	391	06A0AB	xPINIT	466
034314	~xPARTFRAC	466	2E9006	^pisur-2	420
3D4006	^PARTFRACRAT		2E8006	^pisur2	420
0F2007	^PASCAL_NEXTLINE	385	354006	^PIVOTFLOAT	341
2EF94	PASTE.EXT	307	353006	^PIVOTNORM	
393EA	xPATH	466	3C662	xPIX?	466
25EF1	PATHDIR	169	3C638	xPIXOFF	466
228006	^PCAR	346	260DF	PIXOFF	91
04F314	~xPCAR	466	260D5	PIXOFF3	91
0450AB	xPCOEF	466	3C60E	xPIXON	466
00D0AB	xPCONTOUR	466	260E4	PIXON	91
01F0AB	xPCOV	466	260DA	PIXON3	91
02E0E7	~PCunpack		260EE	PIXON?	91
3DC006	^PDer	387	260E9	PIXON?3	92
3C4F5	xPDIM	466	255BF	PixonB	94
3F8006	^PDIV2ext	372	255C4	PixonG1	94
0C6006	^PDivLk	377	255C9	PixonG2	94
011100	*PEEK	478	255BA	PixonW	94
209006	^PEGCD	414	255CE	PixonXor	94
3B477	xPERM	466	3EE9D	xPKT	466
37B006	^PEVAL	345	3FA006	^PLCZ	
0460AB	xPEVAL	466	36FE2	plDRPpZparg	88

OVER# > - xPsi 531

Addr.	Name	Page	Addr.	Name	Page
009314	~xPLOT		0AB006	^PPP	374
1E4006	^PLOTADD	412	2BE006	^PPZ	424
00A314	~xPLOTADD	466	117007	^PPZZ	424
2F35E	PLOTERR		3D0D7	xPR1	467
2F35F	PlotOneMore?		3DBCA	xPREDIV	468
2F0C5	PLOTPREP		3DFDD	xPREDV	467
1E3006	^PLOTSTK		3E01D	xPREDX	467
0A9007	^PLUSAT0	407	3DFFD	xPREDY	467
0AB007	^PLUSATINFTY	407	3ABFD	preFACT	
3A7006	^plusinf	368	2F360	PREMARKON	
2E4006	^PLUSINFext	419	124006	^PREPARext	422
3C392	xPMAX	466	4EB006	^prepvarlist	73
3C372	xPMIN	466	00C314	~xPREVAL	467
0140DE	xPMINI		2D0006	^PREVALext	390
0B3006	^PNFctr	375	25EF2	PrevNonNull	168
103006	^PNMax	379	03E314	~xPREVPRIME	467
010100	xPOKE	478	08376	PREVRAM-WORD	168
3C979	xPOLAR	466	2610C	PrgmEntry?	278
2B682	POLErrorTrap		0C7006	^Prime+	332
2B628	POLKeyUI	223	0C8006	^Prime-	332
2B6CD	POLRestoreUI	223	2F0BC	PRINT	
2B6B4	POLResUI&Err	223	2F361	PrintGrob	
2B4AC	POLSaveUI	223	2F362	PRINTXNLF	
2B542	POLSetUI	223	3D1E7	xPRLCD	467
0D7007	^POLYASYM	380	028FC	PRLG	
02D0DE	xPOLYNOMIAL		38BBF	xPROMPT	467
0CC007	^POLYPARITY	380	2EEF0	PromptIdUtil	48
0D6007	^POLYSYM	380	08B314	~xPROMPTSTO	467
0350DE	xPOP		0BD007	^PROMPTSTO1	167
1DD007	^POPFLAGS	405	0440AB	xPROOT	467
25F19	PopMetaVStack	162	035314	~xPROPFRAC	467
25F1A	PopMetaVStackDROP	160	3D10D	xPRST	467
25F1B	PopVStack	160	3D0F2	xPRSTC	
25F1C	PopVStackAbove	162	3D143	xPRVAR	467
3BB94	xPOS	467	01D0AB	xPSDEV	467
378FA	POS\$	47	3F9006	^PSetSign	372
37906	POS\$REV	47	3E8006	^PSEUDODIV	357
378FA	POSCHR	47	3E7006	^PSEUDOPREP	422
37906	POSCHRREV	47	29A5D	psh	76
376EE	POSCOMP	69	2963E	psh&	76
2E6006	^POSINFext	419	29821	psh1&	77
0DB007	^POSITIFext	401	298C0	psh1&rev	77
2E7006	^POSUNDEFext	420	29693	psh1top&	77
3F7006	^POTENCEext	358	1CB006	^pshder*	390
0380DE	XPOTENTIAL		2973B	pshtop&	76
01B0DE	xPOWEXPAND		29972	pshzer	77
073314	~xPOWMOD	467	29986	pshzerpsharg	88
11E006	^PPow#		0030DE	xPsi	467

Addr.	Name	Page	Addr.	Name	Page
0040DE	xPSI	467	2F367	PUTYMAX	317
0B4006	^PSQFF	375	2F368	PUTYMIN	317
0AD006	^PSqff	374	3EA49	xPVARS	467
036314	~xPTAYL	467	3C5E4	xPVIEW	467
3E5006	^PTAYLext	372	3E283	xPWRFIT	
2F363	PtoR		3C56E	xPX>C	467
4F5006	^PTrim	378	0EB006	^PZadic	399
2C37D	PTYPE>PINFO		2BF006	^PZHSTR	424
37118	PuHiddenVar	170	3701E	pZpargSWAPUn	88
28071	pull	24,77	293F8	$P\{\}N$	71
260F3	PULLCMPEL	64	509006	^QABSext	422
35B006	^PULLEL[S]	342	118006	^QAdd	371
29754	pullpsh1&	77	0E2006	^QAddMod	395
260F8	PULLREALEL	64	508006	^QCONJext	422
28085	pullrev	77	11C006	^QDiv	371
2F076	puretemp?	84	3F5006	^QDiv?	357
2F0AC	PURGALARM%	174	0E5006	^QDivMod	395
08C27	PURGE	167	3EE006	^QDivRem	372
3E87C	xPURGE	467	2B3006	^QGcd	358
0340DE	xPUSH		0E8006	^QGcdExMod	
1DC007	^PUSHFLAGS	405	0E7006	^QGcdMod	396
26265	PushMetaVStack	161	0E6006	^QInvMod	395
25F1D	PushMetaVStack&Drop	161	0F8006	^QIsZero?	334
25F1E	PushVStack	160	0D9006	^QMod	372
25F1F	PushVStack&Clear	160	0DA006	^QMODSYMext	
25F20	PushVStack&Keep	162	111006	^QMul	371
25F21	PushVStack&KeepDROP	162	0E4006	^QMulMod	395
3C0BF	xPUT	467	4FB006	^QNeg	347
35D006	^PUT[]	343	51A006	^QNORMext	422
2F2F5	PUT_FONTE	313	074007	^QPI	418
2F2F7	PUT_STYLE	313	078007	^Qpi	419
31532	PUTAB0		079007	^Qpi%	419
260FD	PUTCMPEL	65	076007	^QpiArry	419
26102	PUTEL	65	077007	^QpiList	419
2626A	PutElemBotVStack	163	075007	^QpiSym	419
2626F	PutElemTopVStack	163	073007	^QpiZ	418
3C139	xPUTI	467	0310AB	xQR	468
2EEF2	PUTINDEP	317	0080DE	xqr	
2EEF3	PUTINDEPLIST	317	0DF006	^QRoot	
075E9	PUTLAM	118	115006	^QSub	371
2B42A	PUTLIST	72	0E3006	^QSubMod	395
2EEF6	PUTPTYPE	317	3E66F	xQUAD	468
26107	PUTREALEL	65	517006	^QUADRANT	38
2EEF4	PUTRES	317	028314	~xQUOT	468
2EEF1	PUTSCALE	318	3D6F6	XQUOTE	468
2F364	PUTSERIAL		0F8007	^QUOTExSIGMA	
2F365	PUTXMAX	316	3EC006	^QUOText	357
2F366	PUTXMIN	316	3F3006	^QUOTOBJext	357

Addr.	Name	Page	Addr.	Name	Page
04B314	~xQXA	468	3DDA9	xRCLSIGMA	468
381006	^QXA	345	2F259	RCLSYSF	175
165006	^QXNDext	421	2F25A	RCLSYSF2	175
11D006	^R15SIMP		2F25B	RCLUSERF	176
400006	^R2SYM	325	2F25C	RCLUSERF2	176
18D006	^R2Zext	328	03F0DE	xRCLVX	468
0701F	R>	128	0C4007	^RCLVX	408
38F01	xR>B	468	505006	^RCONJext	348
3B7ED	xR>C	468	00111	RCS	
3B0AE	xR>D	468	3B6FA	xRCWS	468
3F070	xR>I	469	11B006	^RDIVext	372
2B8E6	R>OBJ	129	3BEEC	xRDM	468
0F5006	^R>Z	327	06FB7	RDROP	129
07012	R@	128	3597F	RDROPCOLA	129
50A006	^RABSext	348	2962A	RDROPFALSE	136
28B006	^RACTOFACext	383	26111	RDUP	129
3B564	xRAD	468	3B401	xRDZ	469
25EF3	RAD?	178	3B819	xRE	469
119006	^RADDext	371	261FC	Re>C%	37
082E3	RAM-WORDNAME	168	33111	real	10
3B3E6	×RAND	468	195006	^REAL?	201
02A0AB	xRANK	468	281006	^REALBICAR	382
0350AB	xRANM	468	36DF3	REALcase	145
113006	^RASOP	372	33233	REALEXT	12
0FD007	^RATSUM	393	528006	^REALLN	349
00114	RBR		09C007	^REALMODE	406
3D393	xRCEQ	468	331A7	REALOB	11
3E6F1	xRCL	468	33733	REALOBOB	17
2B351	Rcl&Do:		25F6D	realPAcode	
29F006	^RCL1IDNT	424	331B1	REALREAL	11
2F196	RCL_CMD	300	33797	REALSTRSTR	17
2F197	RCL_CMD2	300	3320B	REALSYM	11
2EF85	RCL_CMD_DEB	306	25EF4	RECLAIMDISP	273
2EF86	RCL_CMD_FIN	306	3ED22	xRECN	469
2F199	RCL_CMD_MODE	301	2F369	RECORDX&YC%	
2EF87	RCL_CMD_POS	300	0110AB	xRECT	469
26274	RCL_NB_AFF_LGN		0A7007	^RECURMODE	407
26279	RCL_NB_AFF_LGNSTK		3ED56	xRECV	469
3918E	xRCLALARM	468	099006	^REDUCE	384
2F314	RCLALARM%	174	0F6007	^REDUCEMETAPSYST	342
29E006	^RCLALLIDNT		0F5007	^REDUCEMETASYST	342
0C7007	^RCLEPS	408	501006	^xREext	348
3B715	xRCLF	468	048314	~xREF	469
370AF	RclHiddenVar	170	3D7006	^REGCDext	380
3EF79	xRCLKEYS	468	02A314	~xREMAINDER	469
3EA2E	xRCLMENU	468	2F36A	REMAP	
0C2007	^RCLMODULO	408	0130DD	xRENAME	469
0C3007	^RCLPERIOD	408	069004	^RENAME	

Addr.	Name	Page	Addr.	Name	Page
069314	~xREORDER	469	2CD006	^RISCHPF	
23D006	^REORDER	416	2CE006	^RISCHRAT	
071E5	REPEAT	151	15E006	^RIXCext	37
38105	xREPEAT	469	0200AB	xRKF	469
25EF5	REPEATER		0220AB	xRKFERR	469
25EF6	REPEATERCH		0210AB	xRKFSTEP	469
047C7	REPKEY?	206	38E01	xRL	469
25583	Repl	93	38E21	xRLB	469
3B9D2	xREPL	469	4E4006	^RLVARext	397
085D3	REPLACE	167	0C2006	^RmCombNext	376
468006	^REPLACE2BY1	353	112006	^RMULText	371
2579A	REPLACE_MODE		3AEB1	xRND	469
2F2ED	REPLACEALL	309	32F006	^RNDARRY	340
2F2FC	REPLACEALLNOSCREEN	309	313D3	RNDC[B]	
359F7	REQcase	144	30F14	RNDXY	32
35A10	REQcasedrop	144	4FC006	^RNEGext	347
3C41A	xRES	469	3B16C	xRNRM	469
0CA007	^RESETCASCFG	408	3DD18	xROLL	469
2F0A1	RESETDEPTH	107	03325	ROLL	108
35BFF	RESOROMP	100	2B31A	Roll&Do:	
111007	^RESPSHIFTQ	379	29A8F	roll2ND	76
3EAE7	xRESTORE	469	36FCE	roll2top&	76
1D4006	^RESTORECASFLAGS	408	3DD33	xROLLD	469
089002	^RestoreHARDBUFF	234	35FC4	ROLLDROP	108
25F22	RestoreSysFlags	176	35D80	ROLLSWAP	108
37186	RestVarRes	169	36FCE	rolltwotop&	76
10F007	^RESULTANT	379	34FE6	Rom-Word?	
0050DE	xRESULTANT	469	34FCD	ROM-WORD?	100
110007	^RESULTANTLP	379	33715	ROMPANY	16
28187	reversym	107	3319D	rompointer	11
258C7	ReviewKey!	291	08CCC	ROMPTR>#	100
05D0AB	xREVLIST	469	07E99	ROMPTR@	100
0280DE	xREWRITE		06F0AB	xROMUPLOAD	469
46E006	^REWRITEIFINF		3D3CE	xROOT	469
28E006	^RFACT2ext	383	455006	^xroot2expln	367
28D006	^RFACText	383	37D006	^ROOTM2ROOT	385
28F006	^RFACTSTEP3	383	4C9006	^ROOT{}N	396
290006	^RFACTSTEP5	383	3DC71	xROT	469
25F24	RIGHT\$3x6	96	03295	ROT	107
2F36B	RIGHTCOL	279	36761	ROT#+	24
0AF007	^RIGORMODE	407	35E07	ROT#+SWAP	25
500006	^RIMext	348	367B1	ROT#-	24
00D314	~xRISCH	469	36801	ROT#1+	24
2C3006	^risch/		28001	ROT#1+UNROT	24
15C006	^RISCH13	387	35E07	ROT+SWAP	25
2C4006	^rischABS		3574D	ROT2DROP	106, 107
2C2006	^RISCHext		35CA4	ROT2DUP	107
2CF006	^rischlogpart		35CB8	ROTAND	136

Addr.	Name	Page	Addr.	Name	Page
341A8	ROTDROP	107	005007	^RunDoOldMatrix	
34195	ROTDROPSWAP	107	2AB69	RunInApprox	176
3579C	ROTDUP	107	0B954	RunInNewContext	310
35CCC	ROTOVER	107	2A9E9	RunRPN:	198
343BD	ROTROT2DROP	107, 108	2ABF0	RunSafeFlags	176
3416E	ROTSWAP	107	2ABD7	RunSafeFlagsNoError	177
36FBA	ROTUntop&	76	013100	xR~SB	478
03D0AB	xROW+	470	0F2006	^S>Z	328
03C0AB	xROW-	470	0F3006	^S>Z?	328
0370AB	$xROW \rightarrow$	470	008100	$xS \rightarrow H$	478
2F36C	Rows8-15	279	35A5B	SAFE@	166
11F006	^RP#	372	25EF7	SAFE@_HERE	166
123006	^RPext	372	1D3006	^SAFEPURGE	167
070FD	RPIT	138	35A29	SAFESTO	167
070C3	RPITE	138	0000F	sALLOWINTR	
3F218	xRPL>		3C9E5	xSAME	470
0680AB	xrpm		175006	^SAMEMATRIX	338
3885C	xRPN->	465	176006	^SAMEMATSCTYPE	338
2A964	RPNDecomp#Disp	52	1D2006	^SAVECASFLAGS	408
2A944	RPNDecomp#Line	53	261A7	savefmt1	177
2A904	RPNDecomp1Line	52	088002	^SaveHARDBUFF	234
2A984	RPNDecompEcho	54	34D51	SAVELAM	
2A924	RPNDecompEdit	53	2F05E	SaveLastEdit	314
2A9A4	RPNDecompStd1Line	52	25EFB	SaveLastMenu	289
2A9C4	RPNDecompStd1Line32	52	34D58	SAVESTACK	117
26C006	^rpnQOBJext	423	25F23	SaveSysFlags	176
3A200	rpnXROOT		3712C	SaveVarRes	169
38E41	xRR	470	00008	sBEG	
38E61	xRRB	470	00004	sBPOFF	
3DF006	^RRDMext	378	3EE82	xSBRK	470
4FE006	^RREext	348	014100	xSB~B	479
047314	~xrref	470	247006	^SC*MATMOD	
0340AB	xRREF	470	3C4D5	xSCALE	470
223006	^rref	415	2627E	SCANFONT	
078314	~xRREFMOD	470	3E1EF	xSCATRPLOT	470
254006	^RREFMOD		3C9AF	XSCATTER	
0230AB	xRRK	470	0330AB	xSCHUR	470
0240AB	xRRKSTEP	470	3B5BA	xSCI	470
0250AB	xRSBERR	470	329006	^SCL*MAT	339
3B22F	xRSD	470	3E127	xSCLSIGMA	470
368C9	RSKIP	129	3E385	xSCONJ	470
261A2	rstfmt1	178	07D314	~xSCROLL	470
116006	^RSUBext	371	2F36D	SCROLLDOWN	279
34144	RSWAP	129	0C1007	^SCROLLext	281
0400AB	xRSWP	470	2F36E	SCROLLLEFT	280
3E632	xRULES		2F36F	SCROLLRIGHT	280
004002	^RunChooseSimple	244	2F370	SCROLLUP	279
006007	^RunDoNewMatrix		255B0	ScrollVGrob	94
			1		

Addr.	Name	Page	Addr.	Name	Page
3DF32	xSDEV	470	2EE69	SetDA1Temp	275
33157	seco	10	2EE67	SetDA1Valid	275
3FE006	^SECO2CMPCART		2EE94	SetDA23NoCh	276
3FC006	^SECO2CMPext	422	2EE79	SetDA2aBad	276
3FD006	^SECO2CMPPOL		2EE6C	SetDA2aEcho	276
2E107	Seco>Menu	293, 95	2EE73	SetDA2aNoCh	276
461006	^SECOEXEC	421	2EE8A	SetDA2aTemp	275
4D4006	^SECOSQFFext	333,422	2EF98	SetDA2aValid	275
2F2E7	SELECT.FONT	313	2EE7A	SetDA2bBad	276
2F2E1	SELECT.LINE	307	2EE7F	SetDA2bIsEdL	276
2F2E2	SELECT.LINEEND	307	2EE76	SetDA2bNoCh	276
0312B	SEMI	129	2EE6A	SetDA2bTemp	275
34AAD	SEMILOOP	151	2EEA5	SetDA2bTempF	275
3ECB0	xSEND	471	2EE68	SetDA2bValid	275
2F371	SENDACK		2EE93	SetDA2NoCh	276
2F372	SENDEOT		2F37A	SetDA20KTemp	275
2F373	SENDERROR		2EE91	SetDA2Valid	275
2EEBB	SENDLIST	180	2EE7B	SetDA3Bad	276
2F374	SENDNAK		2EE77	SetDA3NoCh	276
2F375	SENDNULLACK		2EE6B	SetDA3Temp	275
2F376	SENDPKT		2EF99	SetDA3Valid	275
2F377	SendSetup		2EEA0	SetDA3ValidF	275
25EF8	SEP\$NL	48	2EE7C	SetDAsNoCh	276
0530AB	xSEQ	471	2EE64	SetDAsTemp	275
01D100	xSERIAL	479	39104	xSETDATE	456
007314	~xSERIES	471	2FFBD	SETDEG	178
419006	^SERIESEXPLN	356	2565F	SetDoStdKeys	224
3ED91	xSERVER	471	2EECE	SetEcma94	
07661	SET		09F007	^SETEXACT	406
25719	SetAlgEntry	278	264D1	SETFIRSTC_0	
26139	SetAlphaAnn	278	2FFEF	SETGRAD	178
25695	SetAppMode	224	07638	SETHASH	100
25671	SetAppSuspOK	224	26283	SetHeader	273
25683	SetBadPOLUI		3714A	SetHiddenRes	170
261AC	setbeep	178	2F37B	SetIOPARErr	158, 181
26116	SETCIRCERR	157	2F458	SETIVLERR	158
09A007	^SETCOMPLEX	406	25EFC	SetKeysNS	290
2F378	SetCursor	304	2613E	SetLeftAnn	277
2611B	SETCURSOR		2F37C	SETLOOPENV	
2F379	SetDA123NoCh	276	04FB6	SETMEMERR	157
2EE65	SetDA12a3NCh	276	0764E	SETMESG	
2EE65	SetDA12a3NoCh	276	2564D	SetNAppKeyOK	224
2EE6F	SetDA12NoCh	276	262E7	SETNONEXTERR	158
2EE71	SetDA12Temp	275	2EEAE	SetNoRollDA2	276
2EE70	SetDA13NoCh	276	25617	SetNUsrKeyOK	208
2EE78	SetDA1Bad	276	0AA007	^SETPLUSAT0	407
2EEAC	SetDAlIsStat	276	04FF2	SETPORTNOTAV	157

Addr.	Name	Page	Addr.	Name	Page
2FFDB	SETRAD	178	0EC007	^SIGNE>	401
2580E	SetRebuild	288	0E6007	^SIGNEATAN	401
26148	SetRightAnn	277	0E4007	^SIGNECOS	401
2F25D	SETROMPART		0EF007	^SIGNEERROR	404
05016	SETROMPERR	157	0E2007	^SIGNEEXP	401
2F37D	SetServMode		0DD007	^SIGNEext	
262D8	SETSIZEERR	158	0EA007	^SIGNELEFT	401
25EFD	SetSomeRow	293	0E1007	^SIGNELN	401
262E2	SETSTACKERR	157	0E9007	^SIGNERIGHT	401
2645E	$\mathtt{setStdEditWid}$	51	0E3007	^SIGNESIN	401
26459	setStdWid	51	0E7007	^SIGNESQRT	401
2614D	SetSysFlag	175	0E5007	^SIGNETAN	401
25EFE	SetThisRow	292	0E0007	^SIGNMOINS	400
39124	xSETTIME	473	0ED007	^SIGNMULText	401
262DD	SETTYPEERR	157	0DF007	^SIGNPLUS	400
26152	SetUserFlag	175	05F314	~xSIGNTAB	471
2622E	SetVStackProtectWord	164	0DE007	^SIGNUNDEF	400
064314	$^{\sim}$ xSEVAL	471	38837	XSILENT'	
3314D	SEVEN	10	0A6007	^SILENTMODE	407
342EA	SEVENROLL	108	4C1006	^SIMP1!	
331B1	SEVENTEEN	11	299006	^SIMP1ext	354
333C3	SEVENTY	14	033314	~xSIMP2	471
333EB	SEVENTYFOUR	14	2AE006	^SIMP3ext	354
3341D	SEVENTYNINE	14	2AF006	^SIMP3LISText	
3B4C9	xSF	471	2B0006	^SIMP3LSTSLOW	
0CD006	^SFactor	330	2A9006	^SIMPext	354
0D80B0	~sFldVal		2AA006	^SIMPEXTOK	
520006	^SHALT		2AD006	^SIMPGCDext	354
3E696	xSHOW	471	29D006	^SIMPIDNT	423
0426A	ShowInvRomp	178	0220DE	xSIMPLIFY	471
0D3007	^SHRINK2ASYM	379	298006	^SIMPLIFY	354
0D1007	^SHRINK2SYM	379	2AAE0	SimplifyExpression	
0D4007	^SHRINKASYM	379	2A8006	^SIMPNDXFext	424
0CF007	^SHRINKEVEN	379	2A0006	^SIMPSYMBS	354
0D2007	^SHRINKSYM	379	2A2006	^SIMPUSERFCN	354
0630AB	XSIDENS	471	29B006	^SIMPVAR	354
3721C	Sig?ErrJmp	158	2A4006	^SIMP	354
0020DE	xSIGMA	471	3A57C	xSIN	471
3DDC4	xSIGMA+	460	428006	^sin/cos	366
3DDEE	xSIGMA-	460	445006	^sin2exp	367
3E0FD	xSIGMACOL	455	444006	^SIN2EXPext	352
301006	^SIGMAEXP2ext	355	42E006	^SIN2ext	351
2FF006	^SIGMAEXPext	355	415006	^sin2tan	366
3E156	xSIGMALINE	460	414006	^SIN2TAN	351
0010DE	xSIGMAVX	471	40C006	^sin2tan/2	365
3A3EE	xSIGN	471	40B006	^SIN2TAN/2	351
237006	^SIGNE	400	42A006	^SIN2TC	351
0DC007	^SIGNE1ext	400	429006	^SIN2TCext	356

Addr.	Name	Page	Addr.	Name	Page
018314	~xSINCOS	471	086314	~xSOLVER	471
303006	^SINEXPA	355	008314	~xSOLVEVX	471
307006	^SINEXPA*	369	1DE006	^SOLVEXFLOAT	412
308006	^SINEXPA*1	369	27E006	^SOLVext	382
305006	^SINEXPA+	368	25EFF	SolvMenuInit	293
306006	^SINEXPA-	368	05E0AB	xSORT	471
52D006	^xSINext	349	2F37E	SORTASLOW	172
3A678	xSINH	471	100006	^SortList	73
449006	^sinh2exp	367	33B55	SPACE\$	43
448006	^SINH2EXPext	352	0AD007	^SPARSEDATA	407
530006	^xSINHext	350	0130AB	xSPHERE	471
0D0007	^SINTEST		2C2D6	SPLITWHERE	
3E331	xSINV	471	0CE006	^SPollard	331
0D3006	^SIsPrime?	331	3A4EF	xSQ	471
1A2006	^siSYMDER		553006	^xSQext	348
33143	SIX	10	2BD006	^SQFF2ext	424
34281	SIXROLL	108	4CC006	^SQFFext	396
331A7	SIXTEEN	11	317EE	SQRF	
3335F	SIXTY	13	3A442	xSQRT	475
333AF	SIXTYEIGHT	14	42F006	^sqrt1-cos^2	366
33387	SIXTYFOUR	13	42D006	^sqrt1-sin^2	366
33369	SIXTYONE	13	41E006	^SQRT2LNEXP	351
3337D	SIXTYTHREE	13	41F006	^sqrt2lnexp	351
33373	SIXTYTWO	13	2C6006	^SQRT_IN?	424
3438D	SIXUNROLL	109	1CC006	^SQRTINVpshd*	390
3BB1F	xSIZE	471	38EC1	xSR	471
0714D	SKIP	131	0280AB	xSRAD	471
35715	skipcola	131	38EE1	xSRB	471
38E81	xSL	471	3EC55	xSRECV	471
38EA1	xSLB	471	0100DD	xSREPL	471
25EFA	SLEEPxcp		00F100	xSREV	478
00C0AB	xSLOPEFIELD	471	2BB21	sscknum2	87
0B1007	^SLOPPY?	408	2F390	xssgeneral	
0B0007	^SLOPPYMODE	407	13E006	^xssSYM#?	
26120	SLOW	172	140006	^xssSYM%	
2B2006	^SLOWGCDext	358	142006	^xssSYM%CH	
2AC006	^SLOWSIMP2L		144006	^xssSYM%T	
297006	^SLVARext	353	3B0006	^xssSYM*	
2BB3A	sncknum2	87	3AC006	^xssSYM+	
3E35B	xSNEG	471	3AE006	^xssSYM-	
00002	SNEGATE		3B2006	^xssSYM/	
2F38F	xsngeneral	77	136006	^xssSYM<=?	
0290AB	xSNRM	471	134006	^xssSYM </td <td></td>	
03F314	~xSOLVE	471	13C006	^xssSYM=?	
219006	^SOLVE1EQ	414	13A006	^xssSYM>=?	
0F7007	^SOLVECRAMER	342	138006	^xssSYM>?	
21A006	^SOLVEMANYEQ	414	3B6006	^xssSYM^	
0F4007	^SOLVEMETASYST	342	152006	^xssSYMAND	

Addr.	Name	Page	Addr.	Name	Page
14C006	^xssSYMCOMB		3E823	xSTO>	167, 472
19F006	^ssSYMDER		2F198	STO_CMD_MODE	301
132006	^xssSYMMAX		2EF8B	STO_CURS_POS	304
130006	^xssSYMMIN		2EF8C	STO_CURS_POS2	304
146006	^xssSYMMOD		2EF8D	STO_CURS_POS3	304
150006	^xssSYMOR		2EF8E	STO_CURS_POS4	304
14E006	^xssSYMPERM		2EF8F	STO_CURS_POS_VIS	304
14A006	^xssSYMRNDXY		2628D	STO_ML_DISP_SIZE	
148006	^xssSYMTRCXY		39164	xSTOALARM	472
154006	^xssSYMXOR		2F3A9	STOALLF	176
12E006	^xssSYMXROOT		2F3AA	STOALLF2	176
25F12	sstDISP	280	2F37F	STOALM	174
26242	StackFontHeight	283	2F066	STOAPPLDATA	
261B1	stackitw		3B749	xSTOF	472
26288	StackLineHeight	283	37104	StoHiddenVar	170
381AB	xSTART	471	2F062	StoIOPAR	181
25F00	StartMenu	292	3EF07	xSTOKEYS	472
2B74F	StartupProc		07D1B	STOLAM	117
38252	xSTARTVAR	459	370006	^STOMAText	344
2EEDA	STATCLST	66	0C6007	^STOMODULO	408
2EEDE	STATMEAN	66	2CA006	^STOPRIMIT	424
2EEDC	STATN	66	2F063	StoPRTPAR	
2EEDB	STATSADD%	66	0240DE	xSTORE	
2EEDD	STATSMAX	66	3DD6E	xSTOSIGMA	472
2EEDF	STATSMIN	66	2F25F	STOSYSF	175
2EEE0	STATSTDEV	66	2F260	STOSYSF2	176
2EEE1	STATTOT	66	356006	^STOSYSText	342
2EEE2	STATVAR	66	2F261	STOUSERF	176
3B5FA	xSTD	472	2F262	STOUSERF2	176
25F01	Std/BoxLabel	95	0400DE	xSTOVX	472
2716D	StdIOPAR	181	0C5007	^STOVX	408
27D7B	StdLabelGrob	90	33125	str	10
25F02	StdMenuKeyLS	290	3BBD9	xSTR>	472
25F03	StdMenuKeyNS	290	2E0F3	Str>Menu	293,95
27A3A	StdPRTPAR		0580AB	xSTREAM	472
3851F	XSTEP	472	2F1AC	StrEdit	
0A3007	^STEPBYSTEP	407	3BE006	^STRICTmetaCOMPARE	370
206006	^STEPIDIV2	413	37ABE	STRIPTAGS	61
3D3AE	xSTEQ	472	37AEB	STRIPTAGS12	62
3EE62	xSTIME	472	33319	STRLIST	13
25F13	stkdecomp\$w	51	33823	STRREALREAL	17
3E739	xSTO	472	38D94	xSTRUCT->	
07D27	STO	167	38D72	xSTRUCT>	
369BE	STO'	130	00001	sTRUNC	
3E4D2	xSTO*	472	222006	^STUDDIV	
3E3AF	xSTO+	472	221006	^STUDMULT	
3E406	xSTO-	472	0160DE	xSTURM	
3E46C	xSTO/	472	0170DE	xSTURMAB	
32100	11010,	1,4	01,000	1.01011111	

Addr.	Name	Page	Addr.	Name	Page
3B6C1	xSTWS	472	39C8B	SWAP>HCOMP	68
2557E	Sub	93	3700A	SWAPCKREF	171
3B8D7	xSUB	472	3635B	SWAPCOLA	131
05733	SUB\$	48	2F073	SWAPcompSWAP	86
2A5CA	SUB\$1#	48	3421A	SWAPDROP	107
35DA8	SUB\$SWAP	48	28989	SWAPDROP#1-	24
05821	SUBCOMP	69	35857	SWAPDROPDUP	107
25597	SubGor	93	35872	SWAPDROPSWAP	107, 108
2612F	SUBGROB	91	374BE	SWAPDROPTRUE	136
2559C	SubGxor	93	3576E	SWAPDUP	107
05815	SUBHXS	57	163006	^SWAPFXND	380
33F006	^submeta	78	3623E	SWAPINCOMP	71
3760D	SubMeta0b	77	3646E	SWAPINDEX@	152
37685	SubMetaOb1	77	36496	SWAPLOOP	151
261B6	subpdcdptch		161006	^SWAPNDXF	379
25592	SubRepl	93	36AFE	SWAPONE	21
0E8007	^SUBSIGNE	401	3457F	SWAPOVER	106, 107
002314	~xSUBST	472	367ED	SWAPOVER#-	24
243006	^SUBTMOD	416	117006	^SWAPRADD	371
06F314	~xSUBTMOD	473	11A006	^SWAPRDIV	372
0F9007	^SUM	393	4FF006	^SWAPRIM	348
3D503	xSUM	460	110006	^SWAPRMULT	371
4D6006	^SUMSQRext	333	4FA006	^SWAPRNEG	348
0FB007	^SUMVX	393	341BA	SWAPROT	107, 108
3DE5A	xSUMX	461	0AC003	^SWAPROWS	344
3DE90	xSUMX2	461	4FD006	^SWAPRRE	348
3DEC6	xSUMXY	461	114006	^SWAPRSUB	371
3DE75	xSUMY	461	2812F	SWAPTRUE	136
3DEAB	xSUMY2	461	36946	SWAPUnDROP	77
10C006	^SUnivar?	380	3695A	SWAPUNNDROP	76
25F04	SuspendOK?	224	58D006	^SWITCHNOTALLOWED	404
02E0AB	xSVD	473	093007	^SWITCHOFF	405
02F0AB	xSVL	473	092007	^SWITCHON	405
03223	SWAP	107	3592B	SWP1+	24,77
3DC20	xSWAP	473	2A7006	^SWPSIMPNDXF	424
357BB	SWAP#-	24	51B006	^SXSQRext	348
3592B	SWAP#1+	24,77	04E314	$^{\sim}$ xSYLVESTER	473
28099	SWAP#1+SWAP	24	384006	^SYLVESTER	346
36829	SWAP#1-	24	3316B	sym	10
280AD	SWAP#1-SWAP	24	50F006	^xSYMABS	
36C22	SWAP%%/	34	53F006	^xSYMACOS	349
362F2	SWAP%>C%	37	550006	^xSYMACOSH	350
35346	SWAP&\$	49	56F006	^xSYMALOG	350
36982	SWAP'	130	514006	^xSYMARG	
368B5	SWAP2DUP	107	53D006	^xSYMASIN	349
36CB8	SWAP3PICK	107	54D006	^xSYMASINH	350
36CE0	SWAP4PICK	109	541006	^xSYMATAN	349
36C90	SWAP4ROLL	107	54A006	^xSYMATANH	350

Addr. Name Page Addr. Name	e Page
33161 symb 10 33675 SYMRE	EAL 16
464006 ^SYMBEXEC 353 2EF26 SYMSF	HOW 88
157006 ^SYMBINCOMP 72,325,359 512006 ^xSYMS	IGN
0546D SYMBN 71,86 538006 ^xSYMS	IN 349
2EFEC symbn 543006 ^xSYMS	INH 350
2BD8C SYMBN: 86 555006 ^xSYMSQ	$_{2}$ 348
2EED9 SYMBNUMSOLVE 552006 ^xSYMSQ	QRT 348
33639 SYMBREAL 16 336B1 SYMSY	YM 16
33657 SYMBUNIT 16 53A006 ^xSYMTA	AN 349
46A006 ^SYMBWHERE 547006 ^xSYMTA	ANH 350
561006 ^xSYMCEIL 350   46F006 ^SYMTA	AYLOR 386
3B8006 ^xSYMCHS 567006 ^xSYMXH	PON 350
26F006 ^SYMCOLCT 26134 SYNTA	AXERR 157
286E7 symcomp <b>86</b> 58F006 ^Sys13	IT 404
507006 ^xSYMCONJ 2F064 Sys@	166
536006 ^xSYMCOS 349 005002 ^sysCF	HOOSE 233
545006 ^xSYMCOSH 349 08D92 SYSCO	ONTEXT 169
55B006 ^xSYMD>R 2EF67 SysD	isplay 274
1A0006 ^SYMDER 2EE5E Sysen	rrorTrap
559006 ^xSYMEXP 364 2F1A3 SysEr	rorTrapAction
29A006 ^SYMEXPAN 354 2EE5F SysEr	rorTrapConfirm
571006 ^xSYMEXPM1 350 39705 xSYSEV	VAL 473
336BB SYMEXT 16 36F79 SysI	TE 146
575006 ^xSYMFACT 350 2EF66 SysMe	enuCheck 292
55F006 ^xSYMFLOOR 350 08D66 SysPt	tr@
565006 ^xSYMFP 350 08DD4 SYSRE	RP? 169
33693 SYMID 16 2F380 SysS	ro <b>167</b>
504006 ^xSYMIM 362 00A0DE xSYST2	2MAT
2A1006 ^SYMINTEGRAL 22A006 ^SYSTE	EM 415
557006 ^xSYMINV 348 80F02 Syste	emFlags
563006 ^xSYMIP 350 26157 Syste	emLevel?
45D006 ^SYMISOL 355006 ^SYST	ext 342
3369D SYMLAM 16 2EED7 SysT	
1DF006 ^SYMLIMIT 412 016100 xS~N	479
	ECOSext 425
	ETANext 425
56D006 ^xSYMLOG 350 061314 ~xTABVA	473
569006 ^xSYMMANT 350 265006 ^TABVA	ALext 418
578006 ^xSYMNOT 350 238006 ^TABVA	_
3366B SYMOB 16 060314 ~xTABVA	
470006 ^SYMPAPRX 33189 TAGGI	
109007 ^SYMPSI 393 336ED TAGGH	EDANY 16
10A007 ^sympsi 37B04 TAGON	
10C007 ^sympsin 0520AB xTAIL	473
10B007 ^SYMPSIN 394 275C6 Take(	
45E006 ^SYMQFORM 3A624 xTAN	473
55D006 ^xSYMR>D 01C0DE xTAN20	
502006 ^xSYMRE 362   442006 ^TAN20	CS2 352

Addr.	Name	Page	Addr.	Name	Page
420006	^TAN2EXP	351	2F158	THISCHAR	300
421006	^tan2exp	366	33125	THREE	10
01F314	~xTAN2SC	473	33319	THREEFIVE	13
427006	^TAN2SC	351	36216	THREE { } N	72
021314	~xTAN2SC2	473	09D006	^THREE{}POLY	377
440006	^TAN2SC2	352	39093	xTICKS	473
43F006	^TAN2SC2ext	357	2F003	Ticks>Date	173
426006	^TAN2SCext	356	2F004	Ticks>Rpt	173
40E006	^TAN2TAN/2	351	2F002	Ticks>TOD	173
40F006	^tan2tan/2	366	34BEF	ticR	128
52E006	^xTANext	349	3905D	XTIME	473
3A70C	xTANH	473	26161	TIMEOUT?	
44D006	^tanh2exp	367	0012E	TIMERCTRL.1	
44C006	^TANH2EXPext	352	0012F	TIMERCTRL.2	
531006	^xTANHext	350	2EED3	TIMESTR	49, 173
1E1006	^TAYLOR0	412	0650AB	xTINC	473
006314	~xTAYLOR0	473	019314	~xTLIN	473
3E6CA	xTAYLR	473	3C6B6	XTLINE	473
00116	TBR		3E97B	xTMENU	473
302006	^TCHEBext	378	26166	TOADISP	272
233006	^TCHEBNOCK	415	2EECF	TOD	172
05B314	~xTCHEBYCHEFF	473	2F381	TOD>t\$	173
01A314	$\sim$ xTCOLLECT	473	2616B	TOGDISP	272
2FE006	^TCOLLECT	355	2F382	TOGGLELINE#3	92
00112	TCS		2577F	TogInsert	302
0640AB	xTDELTA	473	25F2D	TogInsertKey	
2F09A	TempConv	82	2EFA1	TOGLINE	92
3316B	TEN	10	2EFA4	TOGLINE3	92
2E1006	^TESTINFINI	419	2F21B	ToGray	95
02E0DE	xTESTS		33AA7	tok\$	45
26170	TestSysFlag	175	33AB3	tok&	45
26175	TestUserFlag	175	33B85	tok'	44
065314	$\sim$ xTEVAL	473	33BD9	tok*	45
31F006	^TEXPAext	356	33BF1	tok+	45
013314	~xTEXPAND	473	33B91	tok,	44
3C8FA	XTEXT	473	33BFD	tok-	44
37F7F	xTHEN	473	272D9	tok->	44
38B43	XTHENCASE		33B9D	tok.	44
33189	THIRTEEN	11	33BE5	tok/	45
33233	THIRTY	12	33C4D	tok0	44
33283	THIRTYEIGHT	12	33C59	tok1	44
33265	THIRTYFIVE	12	33C65	tok2	45
3325B	THIRTYFOUR	12	33C71	tok3	45
3328D	THIRTYNINE	12	33C7D	tok4	45
3323D	THIRTYONE	12	33C89	tok5	45
33279	THIRTYSEVEN	12	33C95	tok6	45
33251	THIRTYTHREE	12	33CA1	tok7	45
33247	THIRTYTWO	12	33CAD	tok8	44

Addr.	Name	Page	Addr.	Name	Page
26206	tok8cktrior	50	3EE0C	xTRANSIO	474
261BB	tok8trior	50	330006	^TRCARRY	340
33CB9	tok9	44	30F28	TRCXY	32
33BA9	tok;	45	01B314	~xTRIG	474
33AD7	tok<<	44	01C314	~xTRIGCOS	474
33C09	tok=	44	416006	^TRIGext	356
29E99	tok=casedrop	145	082314	~xTRIGO	474
33ACB	tok>>	45	01D314	$^{\sim}$ xTRIGSIN	474
33A6B	tok[	45	01E314	~xTRIGTAN	474
33A51	tok]	45	411006	^TRIGTAN	356
33BCD	tok^	45	4F4006	^TRIMext	378
33B55	tok_	43	4F6006	^TRIMOBJext	398
2D848	tok_g	44	3C084	xTRN	474
2D86D	tok_m	44	3AF3E	xTRNC	474
2D8AD	tok_s	44	2F386	TRPACKETFAIL	
33AEF	tokanglesign	45	03A81	TRUE	135
33C2D	tokCTGROB	45	369D2	TRUE'	130
33C3F	tokCTSTR	45	36540	TRUEFALSE	136
33C21	tokDER	45	36540	TrueFalse	136
33ABF	tokESC	44	27E87	TrueTrue	135
33AE3	tokexponent	44	063314	~xTRUNC	474
33BB5	toklparen	45	471006	^TRUNCDL	386
33B79	tokquote	45	3C99D	xTRUTH	474
33BC1	tokrparen	45	015314	~xTSIMP	474
33A8F	toksharp	45	2B9006	^TSIMP2ext	354
33AFB	tokSIGMA	45	2BB006	^TSIMP3ext	355
33C15	tokSQRT	45	2BA006	^TSIMPext	354
33B61	tokUNKNOWN	45	3125D	TST15	
33A9B	tokuscore	45	391F8	xTSTR	474
33B07	tokWHERE	45	3326F	TTHIRTYSIX	12
33A77	tok{	44	2F034	TURNMENUOFF	273
33A83	tok}	45	2F031	TURNMENUON	273
364E1	toLEN_DO	151	041A7	TurnOff	178
266006	^TOLISText	418	25F05	TurnOffKey	
296A7	top&	76	39456	xTVARS	474
39C006	^top&addt*	360	0470AB	xTVM	474
39D006	^top&addt/	360	0480AB	xTVMBEG	474
36F8D	top&Cr		0490AB	xTVMEND	474
2F383	TOP16	279	04A0AB	xTVMROOT	474
2F384	TOP8	279	3317F	TWELVE	11
2F385	TOPROW	279	331CF	TWENTY	11
07709	TOSRRP	100	3321F	TWENTYEIGHT	11
3DF4D	XTOT	473	33201	TWENTYFIVE	11
06657	TOTEMPOB	171	331F7	TWENTYFOUR	11
35C90	TOTEMPSWAP	171	33229	TWENTYNINE	11
0270AB	XTRACE	474	331D9	TWENTYONE	11
045314	~xTRAN	474	33215	TWENTYSEVEN	11
580006	^TRANSCERROR	403	3320B	TWENTYSIX	11

Addr.	Name	Page	Addr.	Name	Page
331ED	TWENTYTHREE	11	2F099	U>NCQ	82
331E3	TWENTYTWO	11	25F06	UART?	
3311B	TWO	10	2F387	UARTBUFLEN	
09E006	^TWO::POLY	377	25F07	UARTxcp	
36202	$\mathtt{TWO}\{\}\mathtt{N}$	72	38FD7	XUBASE	474
09C006	^TWO{}POLY	377	3900B	XUFACT	474
3BC39	xTYPE	474	0B7006	^UFactor	375
03C64	TYPE	199	0B8006	^UFactor1	375
350D2	TYPEAPLET?	200	0BD006	^UFactorDeg2	376
3513B	TYPEARRY?	199	0140DD	$\mathtt{xUFL1} { ightarrow} \mathtt{MINIF}$	474
350F0	TYPEBINT?	200	2F07C	UM#?	83
352AD	TYPECARRY?	199	2F07D	UM%	83
3503C	TYPECHAR?	200	2F07E	UM%CH	83
03F95	TYPECMP	20	2F07F	UM%T	83
3512C	TYPECMP?	199	2F080	UM*	83
35177	TYPECOL?	200	2D74F	um*	81
3510E	TYPECSTR?	199	2F081	UM+	83
03FDB	TYPEEREL	20	2F082	UM-	83
03FE5	TYPEEXT	20	2D759	um/	81
351B3	TYPEEXT?	200	2F083	UM/	83
35087	TYPEFLASHPTR?	200	2F085	UM<=?	83
350C3	TYPEFONT?	200	2F086	UM </td <td>83</td>	83
114007	^TYPEGAUSSINT?	200,333	2F087	UM=?	83
35186	TYPEGROB?	199	2F088	UM>=?	83
350FF	TYPEHSTR?	199	2F089	UM>?	83
03FA9	TYPEIDNT	20	2F07A	UM>U	82
3504B	TYPEIDNT?	199	2D763	um^	81
194006	^TYPEIDNTLAM?	199	2F08A	UMABS	83
167006	^TYPEIRRQ?	326,421	2D999	UMCEIL	83
03FD1	TYPELAM	20	2F08B	UMCHS	83
350E1	TYPELAM?	199	2F08C	UMCONV	82
03F9F	TYPELIST	20	2F08D	UMCOS	83
35195	TYPELIST?	199	2D777	umEND	81
350B4	TYPELNGCMP?	200	2D985	UMFLOOR	83
350A5	TYPELNGREAL?	200	2D971	UMFP	83
35292	TYPERARRY?	199	2D95D	UMIP	83
03F8B	TYPEREAL	20	2F08E	UMMAX	83
3511D	TYPEREAL?	199	2F08F	UMMIN	83
196006	^TYPEREALZINT?	201	2D76D	umP	81
3514A	TYPEROMP?	200	2D9CB	UMRND	83
20D6F	TYPERRP	20	2F090	UMSI	82
35159	TYPERRP?	200	2D949	UMSIGN	83
03FBD	TYPESYMB	20	2F091	UMSIN	83
35168	TYPESYMB?	199	2F092	UMSQ	83
351A4	TYPETAGGED?	200	2F093	UMSQRT	83
182006	^TYPEZ?	200	2F094	UMTAN	83
35096	TYPEZINT?	200	2D9EE	UMTRC	83
2F07B	U>nbr	82	2F095	UMU>	82

Addr.	Name	Page	Addr.	Name	Page
2F096	UMXROOT	83	404006	^VAL1	423
0310DE	xUNASSIGN		405006	^VAL1M	423
0270DE	XUNASSUME		401006	^VAL2ext	423
2F098	Unbr>U	82	58C006	^VALMUSTBE0	404
262F6	UNCOERCE	31	3FF006	^VALOBJext	423
36BFA	UNCOERCE%%	31	22B006	^VANDERMONDE	415
3F495	UNCOERCE2	31	053314	~xVANDERMONDE	475
29CB9	uncrunch	87	3DF68	xVAR	475
34FA6	undo	117	4C4006	^VAR%	
256AC	UNDO_OFF		45B006	^VAR=LIST	353
256A7	UNDO_ON		4B6006	^VARCOMP!	387
256A2	UNDO_ON?		4B4006	^VARCOMP2!	
2F019	UNIT>\$	82	4B8006	^VARCOMP3!	
33193	unitob	11	4B9006	^VARCOMP31!	
10B006	^Univar?	380	4BA006	^VARCOMP32!	387
25F08	UnLockAlpha	278	4BB006	^VARCOMP33!	
3F249	xUNPICK	474	4B7006	^VARCOMPLN!	
0339E	UNROLL	109	0C4006	^VarFactor	376
29B12	unroll2ND	76	358006	^VARGENext	
3422B	UNROT	108, 108	3943B	xVARS	475
3F22E	XUNROT	474	2F267	VARSIZE	179
343BD	UNROT2DROP	107, 108	37A006	^VBINARYOP	345
35872	UNROTDROP	107, 108	08C314	~xVER	475
35D1C	UNROTDUP	108	579006	^Verbose1	417
360CF	UNROTOVER	108	57A006	^Verbose2	417
341BA	UNROTSWAP	107, 108	57B006	^Verbose3	417
3A6006	^unsignedinf	368	0A5007	^VERBOSEMODE	407
38195	XUNTIL	474	57C006	^VerboseN	417
071C8	UNTIL	151	2EF93	VERIF_SELECTION	306
2F193	UobROT	77	2F388	VerifyTOD	172
2F265	UPDIR	169	1D0006	^VERNUMext	408
39420	xUPDIR	474	00F0AB	xVERSION	475
2F266	USER\$>TAG	61	2F389	VERSTRING	181
463006	^USERFCN?	86	26125	VERYSLOW	172
36F65	UserITE	146	2612A	VERYVERYSLOW	172
25F09	UserKeys?	208	52B006	^xvext	348
23F006	^USERLIDNT	416	3C1006	^vgerxssSYMSUM	394
23E006	^USERLVAR	416	2F2E0	ViewEditGrob	313
371F9	UStackDepth		2F21F	ViewGrobObject	280
3E07D	xUTPC	474	2F19A	ViewLevel1	310
3E0BD	XUTPF	474	2F21D	ViewObject	280
3E09D	XUTPN	474	2F21E	ViewStrObject	280
3E0DD	XUTPT	474	0080DD	xVISIT	475
02F0E7	~UTTYPEEXT0?		00A0DD	xVISITB	475
0110E7	~UTVUNS1Arg		25F0A	VLM	
38F81	xUVAL	474	32B006	^VPMULT	339
3C2AC	xV>	475	0390DE	xVPOTENTIAL	
324006	^VADD	339	342006	^VRRDM	338

Addr.	Name	Page	Addr.	Name	Page
343006	^VRRDMmeta	338	2F296	XEQORDER	168
2EEF7	VSCALE		25F14	XEQPGDIR	168
325006	^VSUB	339	2F297	XEQPURGEPICT	318
3BDB2	XVTYPE	475	2F2A3	XEQRCL	166
379006	^VUNARYOP	345	2F2A7	XEQSETLIB	100
4EA006	^VX!	398	2F2A9	XEQSHOWLS	88
4E9006	^VX>	398	25E79	XEQSTOID	167
4E8006	^VX>LVARext	398	25F0C	XEQStoKey	167
4E2006	^VXINDEP?	397	3BC43	XEQTYPE	199
589006	^VXINDEPERR	404	0700AB	xXGET	475
4E6006	^VXLVARext	397	0BE007	^XGROBext	98
4DB006	^VXXL0	326	33625	XHI	16
4DA006	^VXXL1ext	326	3361B	XHI-1	16
4DD006	^VXXL2	326	021100	xXLIB~	479
4DC006	^VXXL2NR	326	2EF92	XLINE_SIZE?	314
4D7006	^VXXLext	325	3EC35	xXMIT	475
4D9006	^VXXLFext	325	23C006	^XNUM	416
39819	xWAIT	475	067314	~xXNUM	475
2F268	Wait/GetKey	207	03ADA	XOR	136
25F0B	WaitForKey	206	3CB7A	xXOR	475
2A4FC	WaitTbz0		25F0D	XOR\$	50
2D1006	^WARNSING	390	3AD65	xXPON	476
3D605	xWHERE		0710AB	xXPUT	476
380DB	xWHILE	475	068314	~xXQ	476
071EE	WHILE	151	0500AB	xXRECV	476
38A2F	xWHILEEND		3C915	xXRNG	476
2EF61	WINDOW#	281	3A278	xXROOT	476
2F38A	WINDOWBOT?	280	454006	^XROOT2ext	352
2617F	WINDOWCORNER	279	2C8006	^XROOT_IN?	
26184	WINDOWDOWN	279	04F0AB	xXSEND	476
26189	WINDOWLEFT	279	06E0AB	xXSERV	476
2F38B	WINDOWLEFT?	280	51C006	^XSQRext	348
2618E	WINDOWRIGHT	279	0000AB	xXVOL	476
2F38C	WINDOWRIGHT?	280	0030AB	xXXRNG	476
2F38D	WINDOWTOP?	280	3421A	XY > Y	107
26193	WINDOWUP	279	25F0E	XYGROBDISP	91
26198	WINDOWXY	279	341D2	XYZ>	106
0080AB	xWIREFRAME	475	3574D	XYZ>Y	106, 107
370C3	WithHidden	170	3416E	XYZ>YXZ	107
2EFBE	WORDSIZE	57	341A8	XYZ>YZ	107
09A003	^WRAP\$	48	343BD	XYZ>Z	107, 108
08E007	^WRITEMENU	293	3636F	XYZ>ZCOLA	131
390AE	xWSLOG	475	35EF2	XYZ>ZTRUE	136
3E03D	xXCOL	475	35872	XYZ>ZX	107, 108
180006	^XEQ>ARRAY1		3422B	XYZ>ZXY	108, 108
17F006	^XEQ>ARRY	65	34195	XYZ>ZY	107
17C006	^XEQARRY>	65	341BA	XYZ>ZYX	107, 108
2F292	XEQIOBACKUP		341D7	XYZW>	106

Addr.	Name	Page	Addr.	Name	Page
343CF	XYZW>W	108	0C9006	^ZFactor	330
34331	XYZW>WXYZ	108	2B8006	^ZGcd	329
36C90	XYZW>YWZX	107	2B7006	^ZGCDext	329
3423A	XYZW>YZWX	107	51F006	^ZINTSQRT	
31219	Y < = X		0FC006	^ZIsNeg?	334
3E05D	xYCOL	476	0FA006	^ZIsOne?	334
331006	^Yext	65	0D2006	^ZIsPrime?	331
33387	YHI	13	0DD006	^ZMod	329
3C935	xYRNG	476	107006	^ZNLT?	334
00B0AB	xYSLICE	476	105006	^ZNMax	329
0010AB	xYVOL	476	106006	^ZNMin	329
0040AB	xYYRNG	477	255A6	ZoomX	
18A006	^Z2%%	31	255AB	ZoomY	
0EF006	^Z2BIN	22	0D1006	^ZPrime?	331
18E006	^Z2Sext	328	0DC006	^ZQUOText	
265BC	Z<	334	0F0007	^ZSIGNE	401
588006	^Z<0ERR	404	0F1007	^zsigne	402
265D0	Z<=	334	0EE007	^ZSIGNECK	401
265C6	Z<>	334	0E0006	^ZSQRT	329
265C1	Z=	334	0D8006	^ZTrialDiv	332
265B7	Z>	334	0D6006	^ZTrialDiv2	332
19D006	^Z>#	22	0D7006	^ZTrialPrime?	332
587006	^Z>#ERR	404	101006	^ZTrim	329
265CB	Z>=	334	0020AB	xZVOL	477
0F6006	^Z>R	31	189006	^ZZ2C%%ext	37
0F1006	^Z>S	46	25F9A	OLASTOWDOB!	197
0F4006	^Z>ZH	328	25F9A	OLastRomWrd!	197
102006	^ZAbs	329	388006	^1&meta	359
50B006	^ZABS	329	40A006	$1-x^2/1+x^2$	365
10E006	^ZBit?	329	31568	1/X15	
10D006	^ZBits	329	34670	10GETLAM	117
0DE006	^ZDIVext		344A8	10PICK	109
108007	^ZEILBERGER	393	3466B	10PUTLAM	118
33107	ZERO	10	3616F	10UNROLL	109
073C3	ZERO_DO	151	3467A	11GETLAM	118
36568	ZEROFALSE	22	34675	11PUTLAM	118
3709B	ZEROISTOPSTO	152	34684	12GETLAM	118
360BB	ZEROOVER	22	3467F	12PUTLAM	118
040314	~xZEROS	477	3468E	13GETLAM	118
21B006	^ZEROS1EQ	414	34689	13PUTLAM	118
21C006	^ZEROSMANYEQ	415	34698	14GETLAM	118
35E75	ZEROSWAP	22	34693	14PUTLAM	118
37287	ZEROZERO	21	33B13	14SPACES\$	45
37394	ZEROZEROONE	$\frac{-1}{21}$	346A2	15GETLAM	118
373A8	ZEROZEROTWO	$\frac{-1}{21}$	3469D	15PUTLAM	118
37380	ZEROZEROZERO	21	346AC	16GETLAM	118
27D006	^ZFACTO	381	346A7	16PUTLAM	118
05F0AB	xZFACTOR	477	346B6	17GETLAM	118
		-··	1	_ ·	

Addr.	Name	Page	Addr.	Name	Page
346B1	17PUTLAM	118	2D7006	^2DROPTRUE	136
346C0	18GETLAM	118	392006	^2DROPZ0	327
346BB	18PUTLAM	118	031AC	2DUP	106
346CA	19GETLAM	118	3674D	2DUP#+	24
346C5	19PUTLAM	118	358C2	2DUP#<	25
362A2	1_#1-SUB	48	358DC	2DUP#=	25
362A2	1_#1-SUB\$	48	358F8	2DUP#>	25
25E6C	1A/LockA		36CA4	2DUP5ROLL	106
35DEE	1ABNDSWAP	119	36621	2DUPEQ	137
364FF	1GETABND	119	35D30	2DUPSWAP	106, 106
34616	1GETLAM	117	3370B	2EXT	16
2F318	1GETLAMSWP1+	119	3632E	2GETEVAL	119
35F42	1GETSWAP	119	34620	2GETLAM	117
36518	1LAMBIND	116	336E3	2GROB	16
3A4006	^1metainf#	367	336CF	2HXS	16
3A1006	^1metaundef#	367	2F17E	2HXSLIST?	22
2B3A6	<pre>1NULLLAM{ }</pre>	119	155006	^2LAMBIND	116
34611	1PUTLAM	118	33459	2LIST	14
2F180	1REV	30	3A3006	^2metainf#	368
25E6D	1stkdecomp\$w	51	3A0006	^2metaundef#	367
36676	2#0=OR	26	2825E	2NELCOMPDROP	68
30E5B	2%%>%	31	271F4	2NULLLAM{}	119
30E47	2%>%%	31	37087	20b>Seco	73
34B4F	2'RCOLARPITE	138	37046	20VER	109
441006	^2*1-cos/sin	366	4E1006	^2POLYNOMIAL?	397
443006	^2*sin/1+cos	367	3461B	2PUTLAM	118
346D4	20GETLAM	118	343E1	2RDROP	129
346CF	20PUTLAM	118	331B1	2REAL	11
346DE	21GETLAM	118	3570C	2skipcola	131
346D9	21PUTLAM	119	35018	2SWAP	107
346E8	22GETLAM	118	12A006	^2SYMBINCOMP	72,325
346E3	22PUTLAM	119	40D006	$^{2x/1+x^{2}}$	365
346F2	23GETLAM	118	34BBB	3@REVAL	128
346ED	23PUTLAM	119	341D2	3DROP	106
346FC	24GETLAM	118	2D5006	^3DUP	106
346F7	24PUTLAM	119	3462A	3GETLAM	117
34706	25GETLAM	118	156006	^3LAMBIND	116
34701	25PUTLAM	119	27208	<pre>3NULLLAM{ }</pre>	119
34710	26GETLAM	118	34485	3PICK	109
3470B	26PUTLAM	119	36789	3PICK#+	25
3471A	27GETLAM	118	36CCC	3PICK3PICK	109
34715	27PUTLAM	119	360F7	3PICKOVER	109
34BAB	2@REVAL	128	35F1A	3PICKSWAP	109
173006	^2DMATRIX?	339	34625	3PUTLAM	118
03258	2DROP	106	343F3	3RDROP	129
355A5	2DROP00	21	33765	3REAL	17
25E6E	2DropBadKey	207	35703	3skipcola	131
35B32	2DROPFALSE	136	3422B	3UNROLL	108, 108

Addr.	Name	Page	Addr.	Name	Page
341D7	4DROP	106	35369	!!append\$?	49
34634	4GETLAM	117	353F7	!!append\$	49
2B3B7	4NULLLAM{}	119	353EB	!!insert\$	49
3448A	4PICK	109	2F315	!#1+IF <dim-1< td=""><td></td></dim-1<>	
3679D	4PICK#+	25	2F316	!#1-IF>0	
35E20	4PICK#+SWAP	25	25E67	!*triand	50
35E20	4PICK+SWAP	25	25E68	!*trior	50
3610B	4PICKOVER	109	353CD	!append\$	49
35F2E	4PICKSWAP	109	35F6A	!append\$SWAP	49
3462F	4PUTLAM	118	2F191	!DcompWidth	51
3423A	4ROLL	107	3533C	!insert\$	49
3588B	4ROLLDROP	107	263D2	!MATTRNnc	
360E3	4ROLLOVER	108	25F68	!REDIMTEMP	
36043	4ROLLROT	108	25F63	!REDIMUSER	
35F06	4ROLLSWAP	107	03EC2	#*	23
34331	4UNROLL	108	2632D	#*OVF	23
343CF	4UNROLL3DROP	108	03DBC	#+	23
35D44	4UNROLLDUP	108	36851	#+-1	23
36057	4UNROLLROT	108	357FC	#+DUP	24
341DC	5DROP	106	36093	#+OVER	24
3463E	5GETLAM	117	34417	#+PICK	110
3448F	5PICK	109	344DD	#+ROLL	108
34639	5PUTLAM	118	35E39	#+SWAP	24
34257	5ROLL	108	3453D	#+UNROLL	109
358A7	5ROLLDROP	108	03DE0	#-	23
356D5	5skipcola	131	35552	#-#2/	24
34357	5UNROLL	109	36815	#-+1	23
341E8	6DROP	106	3581F	#-DUP	24
34648	6GETLAM	117	360A7	#-OVER	24
34494	6PICK	109	34405	#-PICK	110
34643	6PUTLAM	118	344CB	#-ROLL	108
34281	6ROLL	108	35E4D	#-SWAP	24
3438D	6UNROLL	109	3452B	#-UNROLL	109
341F4	7DROP	106	03EF7	#/	23
34652	7GETLAM	117	03CC7	#0<>	25
34499	7PICK	109	03CA6	#0=	25
3464D	7PUTLAM	118	34A7E	#0=?SEMI	141
342EA	7ROLL	108	36383	#0=?SKIP	141
35BEB	7UNROLL	109	348FC	#0=case	141
3465C	8GETLAM	117	36F15	#0=ITE	141
3449E	8PICK	109	35B96	#0=UNTIL	151
34657	8PUTLAM	118	03DEF	#1+	23
342BB	8ROLL	108	36A13	#1+'	130
3615B	8UNROLL	109	073DB	#1+_ONE_DO	151
34666	9GETLAM	117	35830	#1+DUP	24
344A3	9PICK	109	362CA	#1+LAST\$	48
34661	9PUTLAM	118	35FB0	#1+NDROP	75, 106
28225	9UNROLL	109	34436	#1+PICK	109

Addr.	Name	Page	Addr.	Name	Page
344F2	#1+ROLL	108	352F1	#2=	25
2F222	#1+ROT	24	091B4	#2D541	21
35E61	#1+SWAP	24	03FA9	#2E48	20
34552	#1+UNROLL	109	355FD	#3+	23
03E0E	#1-	23	34465	#3+PICK	109
36851	#1-+	23	2D6006	^#3+ROLL	108
36815	#1-	23	355DF	#3-	23
35E89	#1-1SWAP	24	3A1C2	#304	17
35841	#1-DUP	24	3B9FA	#313	17
3601B	#1-ROT	24	350F5	#37258	21
3628E	#1-SUB\$	48	352E0	#3=	25
28071	#1-SWAP	24,77	35602	#4+	23
281D5	#1-UNROT	24	34474	#4+PICK	110
361EE	#1-{}N	72	355DA	#4-	23
35675	#10*	23	3C11E	#410	17
35620	#10+	23	3B928	#411	17
03880	#102A8	21	33837	#412	17
3375B	#110	17	3BA2D	#414	17
3E17B	#111	17	0803F	#414C1	21
3562A	#12+	23	3B93D	#415	17
2777E	#12F	17	33841	#444	17
3378D	#132	17	3C10F	#450	17
337A1	#134	17	3B952	#451	18
337AB	#135	17	33855	#452	18
337B5	#136	17	3BA18	#454	18
337BF	#137	17	3В913	#455	18
337C9	#138	17	3A12D	#4FF	18
337D3	#139	17	35607	#5+	23
337DD	#13A	17	355D5	#5-	23
337E7	#13B	17	3385F	#510	18
337F1	#13D	17	33869	#511	18
337FB	#13E	17	3BA09	#515	18
3530D	#1<>	25	08ECE	#536A8	21
352FE	#1=	25	33873	#550	18
3639C	#1=?SKIP	142	366FD	#5=	25
36D8A	#1=case	$\overline{142}$	3C8DF	#5B11	20
03E6F	#2*	23	356B8	#6*	23
03E2D	#2+	23	3560C	#6+	23
34451	#2+PICK	109	355D0	#6-	23
34517	#2+ROLL	108	277F6	#605	18
34564	#2+UNROLL	109	27800	#606	18
03E4E	#2-	23	2780A	#607	18
03E8E	#2/	23	27814	#608	18
3380F	#200	17	2781E	#609	18
3C8D0	#2111	20	27828	#60A	18
03FEF	#2614	20	27832	#60B	18
03FEF	#2686	20	27832 2783C	#60C	18
36711	#2<>	25	27846	#60D	18
20111	π4 \/	20	2/010	#00D	10

Addr.	Name	Page	Addr.	Name	Page
2768E	#60E	18	3E759	#8FD	19
27698	#60F	18	3561B	#9+	23
276AC	#611	18	33643	#92	16
276B6	#612	18	3364D	#9A	16
276C0	#613	18	3EAFB	#9F	16
276CA	#614	18	3E7E9	#9F1	19
0657E	#61441	21	3E743	#9FD	19
276D4	#615	18	25F72	#:>\$	46
276DE	#616	18	03CE4	#<	25
276E8	#617	18	366BC	#<3	25
27792	#618	18	03D4E	#<>	25
2779C	#619	18	36D9E	#<>case	141
277A6	#61A	18	36D76	# <case< td=""><td>141</td></case<>	141
277B0	#61B	18	36F29	# <ite< td=""><td>141</td></ite<>	141
277BA	#61C	18	03D19	#=	25
277C4	#61D	18	363B5	#=?SKIP	140
277CE	#61E	18	348D2	#=case	141
277D8	#61F	18	34939	#=casedrop	141
277E2	#620	18	36590	#=casedrpfls	141
277EC	#621	18	35C54	#=ITE	141
276F2	#622	18	37752	#=POSCOMP	69
276FC	#623	19	03D83	#>	25
27706	#624	19	25F77	#>\$	46
27710	#628	19	36739	#>1	25
2771A	#629	19	36DB2	#>2case	142
27724	#62A	19	363E2	#>?SKIP	140
2772E	#62B	19	36DCB	#>case	141
27738	#62C	19	05A75	#>CHR	47
27742	#62D	19	059CC	#>HXS	56
27788	#62E	19	36F3D	#>ITE	141
35611	#7+	23	07E50	#>ROMPTR	100
33891	#700	19	0EE006	^#>Z	327
3C17A	#710	19	3CD21	x#?	476
3C16B	#750	19	3373D	#_102	17
08DF7	#7FF	19	2774C	#A01	19
3569B	#8*	23	27756	#A02	19
35616	#8+	23	27760	#A04	19
27878	#800	19	2776A	#A05	19
3B976	#822	19	27774	#A06	19
3C83C	#82C	19	338CD	#A11	19
3B967	#855	19	3D50D	#A110	20
3C81E	#85C	19	338D7	#A12	19
3389B	#861	19	338E1	#A1A	19
338A5	#862	19	3D52B	#A1A0	20
338AF	#865	19	3367F	#A2	16
338B9	#86E	19	338EB	#A21	19
3362F	#8F	16	338F5	#A22	20
3E7FF	#8F1	19	338FF	#A2A	20

Addr.	Name	Page	Addr.	Name	Page
39E6B	#A4	16	36851	\$1-+	23
33689	#A5	16	25F7C	\$>GROB	96
33909	#A61	20	25F81	\$>grob	96
33913	#A62	20	25F86	\$>GROBCR	96
3391D	#A65	20	25F8B	\$>grobCR	96
33927	#A6E	20	05F0B3	~\$>grobOrGROB	96
03826	#A8241	21	05B15	\$>ID	116
336A7	#A9	16	3402C	\$_''	44
33931	#AA1	20	34056	\$_2DQ	44
3D51C	#AA10	20	3403A	\$_::	44
3393B	#AA2	20	34002	\$_<<>>	44
33945	#AAA	20	3401E	\$_[]	44
2C4D2	#AAA0	20	34064	\$_ECHO	44
3BD4C	#AF	16	34076	\$_EXIT	44
03EB1	#AND	25	340B4	\$_GRAD	44
39277	#B437D	21	34048	<pre>\$_LRParens</pre>	44
38275	#BB	16	33FD2	\$_R<<	44
3B7AD	#BBBB	20	33FE2	\$_R <z< td=""><td>44</td></z<>	44
3394F	#C06	20	340A4	\$_RAD	44
33959	#C07	20	34088	<pre>\$_Undefined</pre>	44
33963	#C08	20	33FF2	\$_XYZ	44
33977	#C0B	20	34010	\$_{}	44
3B904	#C22	20	33B45	\$DER	45
3C800	#C2C	20	3B251	x%	476
3B8F5	#C55	20	30385	%% <b>*</b>	33
3C7E2	#C5C	20	3602F	%%*ROT	33
3E7DA	#C8	16	35EDE	%%*SWAP	33
33981	#CAlarmErr	20	36C7C	%%*UNROT	33
3BD65	#CF	16	3032E	%%+	33
08F1F	#D6A8	20	3033A	% % <b>–</b>	33
038DC	#E13A8	21	2FBE5	%%.1	30
33995	#EXITERR	21	30DC8	%%.4	30
07C007	^#FACT	330	2FBFF	%%.5	30
38266	#FFFF	21	303D3	%% <i>/</i>	34
3736E	#FIVE#FOUR	21	36BE6	%%/>%	34
3551D	#MAX	25	2FB49	880	30
35511	#MIN	25	30112	%%O<	35
33783	#NoRoomForSt	17	301F6	%%0<=	35
37315	#ONE#27	21	301A6	%%O<>	35
03DC7	#PUSHA-		30145	%%O=	35
33747	#SyntaxErr	17	30173	%%0>	35
3735C	#THREE#FOUR	21	301CE	%%0>=	35
3734A	#TWO#FOUR	21	2FB63	%%1	30
37328	#TWO#ONE	21	3047D	%%1/	34
3733A	#TWO#TWO	21	2FC19	<b>%%10</b>	30
37294	#ZERO#ONE	21	30CC7	%%12	30
37305	#ZERO#SEVEN	21	2FB7D	882	30
39C9F	\$&ob	53	27A89	%%2PI	30

#A4 - %23 553

Addr.	Name	Page	Addr.	Name	Page
2FB97	883	30	2FA33	%−3	28
2FBB1	<b>%%4</b>	30	2FA48	% <b>-</b> 4	28
2FBCB	%%5	30	2FA5D	%-5	28
30CEB	%%60	30	2FA72	%-6	28
30BEA	887	30	2FA87	%-7	28
3020A	%%<	35	2FA9C	%-8	28
30296	%%<=	35	2FAB1	%-9	28
3026A	%%>	35	2FB0A	%-MAXREAL	28
2FF9B	%%>%	31	2FB34	%-MINREAL	28
30280	%%>=	35	27118	%.1	28
261CF	%%>C%	37	2712D	%.15	
3044A	% <sup>8</sup> ^	34	339BE	%.5	28
302DB	%%ABS	34	303E9	% /	32
306F3	%%ACOSRAD	34	2F937	%0	28
3073A	%%ANGLE	34	30123	%0<	35
30757	%%ANGLEDEG	34	301BA	%0<>	35, 135
30767	%%ANGLERAD	34	30156	%O=	35
306C3	%%ASINRAD	34	2B149	%0=case	142
302FB	%%CHS	34	30184	%0>	35
30642	%%COS	34	301E2	%0>=	35
30653	%%COSDEG	34	2F94C	%1	28
307B2	%%COSH	34	26F36	%1+	31
30663	%%COSRAD	34	26F4A	%1-	31
30507	%%EXP	34	270EE	%1.8	28
30984	%%FLOOR	34	3049A	%1/	32
30912	%%H>HMS	172	339E8	%10	28
30984	%%INT	34	35C18	%10*	31
30546	%%LN	34	27E5D	%100	29
3057F	%%LNP1	34	2FCE6	%11	28
300C7	%%MAX	34	2FCFB	%12	28
30EB0	%%P>R	34	2FC7D	%1200	29
2FADB	%%PI	30	2FD10	%13	29
11C007	^%%PSI	394	2FD10 2FD25	%14	29
30E83	%%R>P	34	2FD23 2FD3A	%15	29
30E63	%%SIN	34	2FCD1	%15360	29
30602	%%SINDEG	34	2FCD1 2FD4F	%15500 %16	29
30780	%%SINH	34	2FD4F 2FD64	%17	29
30780	%%SINH %%SINRAD	34	2FD04 2FD79	%18	29
304D5		34			29
30405	%%SQRT	34	339FD	%180 %10	29
	%%TANRAD %*	31	2FD8E 2B1A3	%19 %1-gaga	142
303A7	*			%1=case	
3035F	%+ %+CNAD	31	4CE006	^%1TWO	396 28
25E69	%+SWAP	31	2F961	%2 %20	
3036C	%- °. г	31	2FDA3	%20 %200	29
339D3	%5	28	33A12	%200 %21	29
2FA09	%-1	28	2FDB8	%21 	29
2B289	%-1=case	143	2FDCD	%22	29
2FA1E	%−2	28	2FDE2	%23	29

Addr.	Name	Page	Addr.	Name	Page
2FDF7	%24	29	30723	%ANGLE	32
2FC92	%2400	29	306AC	%ASIN	32
2FE0C	%25	29	307EB	%ASINH	32
2FE21	%26	29	3070C	%ATAN	32
2FE36	%27	29	30811	%ATANH	32
2FE4B	%28	29	3095E	%CEIL	32
2FE60	%29	29	3B362	x%CH	455
2B20C	%2=case	143	3041B	%CH	33
2F976	%3	28	3030B	%CHS	32
2FE75	%30	29	3084D	%COMB	33
2FE8A	%31	29	3062B	%COS	32
2FE9F	%32	29	307C5	%COSH	32
2FEB4	%33	29	3000D	%D>R	33
2FEC9	%34	29	339A9	%e	28
2FEDE	%35	29	3051A	%EXP	32
33A27	%360	29	3052D	%EXPM1	32
2F98B	%4	28	30824	%EXPONENT	32
33A3C	%400	29	30AAF	%FACT	33
2FCA7	%4800	29	30971	%FLOOR	32
2F9A0	%5	28	30938	%FP	32
2F9B5	%6	28	3008B	%HMS+	172
2F9CA	%7	28	300B3	%HMS-	172
2F9DF	%8	28	30077	%HMS>	172
27103	%80	29	3094B	%IP	32
2F9F4	%9	28	2B3FD	%IP>#	31
2FCBC	%9600	29	30559	%LN	32
3025C	%<	35	30592	%LNP1	32
302A1	%<=	35	3056C	%LOG	32
302B7	%<>	35	3031B	%MANTISSA	32
302AC	%=	35	300E0	%MAX	33
30275	%>	35	35DBC	%MAXorder	33
2EFCB	%>#	56	2FAF5	%MAXREAL	30
2FFAC	%>%%	31	300F9	%MIN	33
30346	%>%%-	31	2FB1F	%MINREAL	28
30489	%>%%1/	32	305C7	%MOD	32
30746	%>%%ANGLE	32	30837	%NFACT	33
304E1	%>%%SQRT	32	3046C	%NROOT	33
35ECA	%>%%SWAP	31	303B4	%OF	33
3028B	%>=	35	30860	%PERM	33
05C27	%>C%	37	2FAC6	%PI	28
3005E	%>HMS	172	30EA6	%POL>%REC	33
2F223	%>TAG	61	30040	%R>D	33
3045B	%^	32	309AD	%RAN	33
302EB	%ABS	32	30A2F	%RANDOMIZE	33
262EC	%ABSCOERCE	22	30E79	%REC>%POL	33
306DC	%ACOS	32	302C2	%SGN	32
307FE	%ACOSH	32	305DA	%SIN	32
305A5	%ALOG	32	30799	%SINH	32
JUJAJ	0111100	92	30179	ODINI	52

%24 - >TAG 555

Addr.	Name	Page	Addr.	Name	Page
30EDD	%SPH>%REC	33	39CFC	х-	476
304F4	%SQRT	32	3DA3E	x->Q	468
3B2DC	x%T	474	3DA63	x->QPI	468
303F6	%T	33	3EFB1	x->TAG	473
3067C	%TAN	32	126006	^x-ext	347
307D8	%TANH	32	39F49	x/	476
4EA61	%TICKSday	29	129006	^x/ext	347
4EA4C	%TICKShour	29	05445	::N	71
4EA37	%TICKSmin	29	3631A	::NEVAL	74
4EA22	%TICKSsec	29	09F006	^::POLY	377
4EA76	%TICKSweek	30	3F053	x;	
05193	&\$	49	3CE42	x<	476
36FF6	&\$SWAP	49	389B9	x<<	
0521F	&COMP	68	3CF80	x<=?	476
0518A	&HXS	57	27F47	<delkey< td=""><td>314</td></delkey<>	314
389EF	x'		27EAF	<skipkey< td=""><td>314</td></skipkey<>	314
06E97	1	130	38D83	x <struct< td=""><td></td></struct<>	
25E6A	'DoBadKey	130	398B9	x=	476
25E6B	'DoBadKeyT	130	3CBF6	x==	476
2B90B	'DROPFALSE	130	128006	^x=ext	350
3619E	'ERRJMP	130	3CEE1	x>	476
27B43	'IDFUNCTION	131	3D01F	x>=?	476
27B7F	'IDPARAMETER	131	389D4	x>>	1.0
27B6B	'IDPOLAR	131	38999	x>>ABND	
27155	'IDX	116	3BE9B	x>ARRY	454
2F350	'LamKPSto	110	27F9A	>DelKey	314
36A8B	'LAMLNAMESTO	168	25F15	>FONT	283
36306	'NOP	130	3C8A1	x>GROB	460
06EEB	'R	128	0525B	>H\$	49
36A27	'R'R	128	0523B	>HCOMP	68
29ED0	'Rapndit	131	3B0EC	x>HMS	461
06F66	'REVAL	128	0A2006	^>HPOLY	378
36A4A	'RRDROP	128	0A4006	^>HPOLYN	378
354CB	'RSaveRomWrd	197	25F90	>LANGUAGE	179
354CB	'RSAVEWORD	197	37C06	>LASTRAM-WORD	
29786	'RSWAP#1+	24	3C881	x>LCD	463
36BBE	'X*	130	3B7D2	x>LIST	464
36BD2	'xDER	130	2620B	>MINIFONT	283
36AA4	'xDEREQ	131	39785	x>NUM	465
2EE006	-	420	0A7006		377
2EC006	^'xPI	$\frac{420}{420}$	4F8006	^>POLYTRIM	399
39DE8 127006	x* ^+	476	06F9F	>R	128 $52$
	^x*ext	347 470	25E6F	>Review\$	
3C444	x*H	470	0EB007	^>SIGNE	401
3C464	x*W	470	27EFB	>SkipKey	314
39B58	X+	476	3BBBE	x>STR	472
125006	^x+ext	347	052EE	>T\$	49
073A5	+L00P	151	05E81	>TAG	61

Addr.	Name	Page	Addr.	Name	Page
052FA	>TCOMP	68	35F56	?SWAP	138
0A1006	^>TPOLY	377	35F97	?SWAPDROP	138
0A3006	^>TPOLYN	378	293A3	?symcomp	86
38FB5	x>UNIT	474	2ACB0	?TogU/LCase	
3C2D6	x>V2	475	0797B	@	166
3C30A	x>V3	475	07943	@LAM	117
4F1006	^>VARLIST		16B006	^[]TO{}	338
089314	~x?		002100	$x{ ightarrow} A$	478
35A88	?>ROMPTR	100	00B100	$x{ ightarrow} ALG$	478
25F9F	?ACCPTR>		006100	$x \rightarrow CD$	478
25E70	?ATTN_QUIT	207	0380AB	$x{ ightarrow} COL$	455
25E70	?ATTNQUIT	207	03A0AB	$x{ ightarrow}$ DIAG	457
25E71	?BlinkCursor	179	0020DD	$x{ ightarrow} FONT$	459
361C6	?CARCOMP	68	000100	$x \rightarrow H$	478
25E72	?CaseKeyDef	142	0040DD	$x{ ightarrow} HEADER$	461
25E73	?CaseRomptr@	142	06C0AB	$x{ ightarrow} KEYTIME$	463
25E74	?ClrAlg	278	0000DD	$x{\rightarrow} \texttt{LANGUAGE}$	463
25E75	?ClrAlgSetPr	278	00A100	$x{ ightarrow} LST$	478
2F19F	?DispCommandLine	312,274	0110DD	$x \rightarrow MINIFONT$	464
2DFCC	?DispMenu	292, 274	0060DD	$x \rightarrow NDISP$	465
2C341	?DispStack	274	00C100	$x \rightarrow PRG$	478
2C311	?DispStatus	274	00E100	$x \rightarrow RAM$	478
2E5006	^?ext	419	0360AB	$x \rightarrow ROW$	470
2EE5D	?FlashAlert	178	020100	x→S2	479
39332	?GETMSG	157	3D202	$x\partial$	457
34A46	?GOTO	129	3D434	x∫	462
25E76	?Key>UKey0b		0550AB	x $\Delta$ LIST	460
3705A	?Ob>Seco	73	0590AB	$\mathtt{x}\Sigma\mathtt{LIST}$	461
25E77	?OKINALG	201	08A314	$\sim$ x $\infty$	462
25E78	?PURGE_HERE	167	05A0AB	$x\Pi$ LIST	466
35AAB	?ROMPTR>	100	3A097	x^	477
34AA1	?SEMI	137	12B006	^x^ext	347
3692D	?SEMIDROP	138	05459	{ }N	71
0712A	?SKIP	138	0A0006	^{}POLY	377
35DDA	?SKIPSWAP	138	16A006	^{}TO[]	338
2EF73	?Space/Go>	311	3D56B	x	460
25E79	?STO_HERE	167			

## **Appendix G Entries sorted by Address**

The entries in this index are 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. Note that for technical reasons, the page number given may be off by one for a few percent of the entries. If the page reference is 167, the entry may actually be the first entry on page 168.

Addr.	Name	Page	Addr.	Name	Page
00001	sTRUNC		028FC	PRLG	_
00002	SNEGATE		0312B	SEMI	129
00003	DZP		0314C	DEPTH	107
00004	sBPOFF		03188	DUP	106
80000	sBEG		031AC	2DUP	106
0000F	sALLOWINTR		031D9	NDUP	106
000FF	allkeys		03223	SWAP	107
00110	IOC		03244	DROP	106
00111	RCS		03258	2DROP	106
00112	TCS		0326E	NDROP	75, 106
00113	CRER		03295	ROT	107
00114	RBR		032C2	OVER	109
00116	TBR		032E2	PICK	109
0011A	IRC		03325	ROLL	108
0011F	IRAM@		0339E	UNROLL	109
0012E	TIMERCTRL.1		0371D	GETATELN	64
0012F	TIMERCTRL.2		03826	#A8241	21
0020F	OUTCINRTN		03880	#102A8	21
00A0E	addrKEYSTATE		038DC	#E13A8	21
00C0D	kermsendmsg		039EF	ECUSER	
00C0E	kermrecvmsg		03A81	TRUE	135
00C10	kermpktmsg		03AC0	FALSE	136
01118	LowBat?	178	03ADA	XOR	136
01661	addr0Rghost		03AF2	NOT	136
026FE	DOMINIFONT		03B2E	EQ	137

Addr.	Name	Page	Addr.	Name	Page
03B46	AND	136	04D87	JstGetTHEMESG	157
03B75	OR	136	04D87	JstGETTHEMSG	157
03B97	EQUAL	137	04E07	GETEXITMSG	156
03C64	TYPE	199	04E37	EXITMSGSTO	156
03CA6	#0=	25	04E5E	ERRSET	157
03CC7	#0<>	25	04E66	addrTEMPENV	
03CE4	#<	25	04EA4	ABORT	156
03D19	#=	25	04EB8	ERRTRAP	157
03D4E	#<>	25	04ED1	ERRJMP	156
03D83	#>	25	04FB6	SETMEMERR	157
03DBC	#+	23	04FF2	SETPORTNOTAV	157
03DC7	#PUSHA-		05016	SETROMPERR	157
03DE0	#-	23	05040	ATTNFLG@	207
03DEF	#1+	23	05068	ATTNFLGCLR	207
03E0E	#1-	23	05089	CARCOMP	68
03E2D	#2+	23	050ED	CAR\$	47
03E4E	#2-	23	05149	Loop	
03E6F	#2*	23	05153	CDRCOMP	68
03E8E	#2/	23	0516C	CDR\$	47
03EB1	#AND	25	0518A	&HXS	57
03EC2	#*	23	05193	&\$	49
03EF7	#/	23	0521F	&COMP	68
03F8B	TYPEREAL	20	0525B	>H\$	49
03F95	TYPECMP	20	052C6	>HCOMP	68
03F9F	TYPELIST	20	052EE	>T\$	49
03FA9	TYPEIDNT	20	052FA	>TCOMP	68
03FA9	#2E48	20	05445	::N	71
03FBD	TYPESYMB	20	05459	{ }N	71
03FD1	TYPELAM	20	0546D	SYMBN	71,86
03FDB	TYPEEREL	20	05481	EXTN	81, 71
03FE5	TYPEEXT	20	054AF	INNERCOMP	71
03FEF	#2614	20	0556F	NULL\$?	55
03FF9	#2686	20	055B7	NULLCOMP?	68
041A7	TurnOff	178	055D5	NULLHXS	57
041ED	DEEPSLEEP	178	055DF	NULL\$	43
0426A	ShowInvRomp	178	055E9	$\mathtt{NULL}\{\ \}$	72
04708	CHECKKEY	205	055FD	NULL::	73
04714	GETTOUCH	206	05616	LENHXS	57
047C7	REPKEY?	206	05622	OVERLEN\$	47
047CF	adrDISABLE_K		05636	LEN\$	47
047DD	adrKEYBUFFER		0567B	LENCOMP	68
04A0B	GETPROC	288	056B6	NTHELCOMP	68
04A41	GETDF	288	05733	SUB\$	48
04CE6	ERROR@	156	05815	SUBHXS	57
04D0E	ERRORSTO	156	05821	SUBCOMP	69
04D33	ERRORCLR	156	05902	OSIZE	178
04D3E	DROPNULL\$	46	05944	OCRC	179
04D64	GETTHEMESG	157	059CC	#>HXS	56

03B46 - 25579 559

Addr.	Name	Page	Addr.	Name	Page
05A03	HXS>#	22	073CE	ONE_DO	151
05A51	CHR>#	22	073DB	#1+_ONE_DO	151
05A75	#>CHR	47	073F7	DO	151
05AB3	CHANGETYPE	179	07497	ABND	117
05B15	\$>ID	116	074D0	BIND	116
05BE9	ID>\$	46	074E4	DOBIND	116
05C27	%>C%	37	075A5	GETLAM	117
05D2C	C%>%	31	075E9	PUTLAM	118
05DBC	C%%>%%	37	07638	SETHASH	100
05E81	>TAG	61	0764E	SETMESG	
05F2E	ID>TAG	61	07661	SET	
05F42	GARBAGE	178	076AE	OFFSRRP	100
05F61	MEM	178	07709	TOSRRP	100
0657E	#61441	21	07943	@LAM	117
06657	TOTEMPOB	171	0797B	@	166
06B4E	INTEMNOTREF?	171	07D1B	STOLAM	117
06E8E	NOP	128	07D27	STO	167
06E97	1	130	07E50	#>ROMPTR	100
06EEB	'R	128	07E99	ROMPTR@	100
06F66	'REVAL	128	0803F	#414C1	21
06F8E	EVAL	128	081D9	BAKNAME	101
06F9F	>R	128	082E3	RAM-WORDNAME	168
06FB7	RDROP	129	08309	MYRAMROMPAIR	
06FD1	COLA	131	08326	LASTRAM-WORD	168
07012	R@	128	08376	PREVRAM-WORD	168
0701F	R>	128	085D3	REPLACE	167
070C3	RPITE	138	08696	CREATE	167
070FD	RPIT	138	08C27	PURGE	167
0712A	?SKIP	138	08CCC	ROMPTR>#	100
0712A	NOT_IT	138	08D08	CONTEXT!	169
0714D	SKIP	131	08D5A	CONTEXT@	169
0716B	IDUP	128, 150	08D66	SysPtr@	
071A2	BEGIN	150	08D92	HOMEDIR	169
071AB	AGAIN	151	08D92	SYSCONTEXT	169
071C8	UNTIL	151	08DD4	SYSRRP?	169
071E5	REPEAT	151	08DF7	#7FF	19
071EE	WHILE	151	08ECE	#536A8	21
07221	INDEX@	151	08F1F	#D6A8	20
07249	ISTOP@	152	0905F	BAK>OB	101
07258	JINDEX@	152	091B4	#2D541	21
07264	JSTOP@	152	092DB	InitEnab	
07270	INDEXSTO	152	0B954	RunInNewContext	310
07295	ISTOPSTO	152	20D6F	TYPERRP	20
072AD	JINDEXSTO	152	25565	LineW	93
072C2	JSTOPSTO	152	2556A	LineB	93
07334	LOOP	151	2556F	LineG1	93
073A5	+LOOP	151	25574	LineG2	93
073C3	ZERO_DO	151	25579	LineXor	93

Addr.	Name	Page	Addr.	Name	Page
2557E	Sub	93	25790	INSERT?	302
25583	Repl	93	25795	INSERT_MODE	302
25588	Gor	93	2579A	REPLACE_MODE	
2558D	Gxor	94	257A2	EditLExists?	300
25592	SubRepl	93	257BE	ClrNewEditL	
25597	SubGor	93	257E2	NoIgnoreAlm	
2559C	SubGxor	93	2580E	SetRebuild	288
255A1	Grey?	94	25845	MenuDef@	289
255A6	ZoomX		25863	MenuRowAct!	289
255AB	ZoomY		25877	LabelDef!	290
255B0	ScrollVGrob	94	25886	DoLabel	293, 95
255B5	Distance	95	2588B	MenuKeyNS!	290
255BA	PixonW	94	25890	MenuKeyNS@	290
255BF	PixonB	94	2589A	DoMenuKeyNS	293
255C4	PixonG1	94	2589F	MenuKeyLS!	290
255C9	PixonG2	94	258B3	MenuKeyRS!	290
255CE	PixonXor	94	258C7	ReviewKey!	291
255D3	FBoxW	94	258EF	ExitAction!	291
255D3	FBoxG1	94	25908	LastMenuDef!	289
255D8	FBoxG2	94	2590D	LastMenuDef@	289
255DD	FBoxB	94	2593F	KeyOb0	
255E2	FBoxXor	94	25949	KeyOb!	
255E7	LBoxW	94	2594E	KeyOb@	
255EC	LBoxG1	94	25967	GetUserKeys	208
255F1	LBoxG2	94	2597B	CtlAlarm!	
255F6	LBoxB	94	25980	CtlAlarm@	
255FB	LBoxXor	94	25E67	!*triand	50
25617	SetNUsrKeyOK	208	25E68	!*trior	50
2561C	ClrNUsrKeyOK	208	25E69	%+SWAP	31
25621	NonUsrKeyOK?	208	25E6A	'DoBadKey	130
25636	HISTON?		25E6B	'DoBadKeyT	130
2564D	SetNAppKeyOK	224	25E6C	1A/LockA	
2565A	DoStdKeys?	224	25E6D	1stkdecomp\$w	51
2565F	SetDoStdKeys	224	25E6E	2DropBadKey	207
25671	SetAppSuspOK	224	25E6F	>Review\$	52
25676	ClrAppSuspOK	224	25E70	?ATTNQUIT	207
25683	SetBadPOLUI		25E70	?ATTN_QUIT	207
25690	AppMode?	224	25E71	?BlinkCursor	179
25695	SetAppMode	224	25E72	?CaseKeyDef	142
2569A	ClrAppMode	224	25E73	?CaseRomptr@	142
256A2	UNDO_ON?		25E74	?ClrAlg	278
256A7	UNDO_ON		25E75	?ClrAlgSetPr	278
256AC	UNDO_OFF		25E76	?Key>UKey0b	
256BE	NOBLINK	179	25E77	?OKINALG	201
256EA	AlgEntry?	278	25E78	?PURGE_HERE	167
25719	SetAlgEntry	278	25E79	XEQSTOID	167
2571E	ClrAlgEntry	278	25E79	?STO_HERE	167
2577F	TogInsert	302	25E7A	ALARMxcp	

2557E - 25EDC 561

Addr.	Name	Page	Addr.	Name	Page
25E7B	ALGeq?		25EAD	DISPSTATUS2	282
25E7C	AND\$	49	25EAE	DO#EXIT	156
25E7D	ATTNxcp		25EAF	DO\$EXIT	156
25E7E	BLANKIT	277	25EB0	DO%EXIT	156
25E7F	Box/StdLabel	95	25EB1	DO>STR	53
25E80	Box/StdLbl:	95	25EB2	DOBEEP	178
25E81	C%1/	37	25EB3	DOCHR	46
25E82	C%>%%	37	25EB4	DODISP	281
25E83	C%>%%SWAP	37	25EB5	DORCLE	318
25E84	C%ABS	37	25EB6	DOSTOE	318
25E85	C%ACOS	38	25EB7	DOSTR>	50
25E86	C%ACOSH	38	25EB8	DOTVARS%	168
25E87	C%ALOG	38	25EB9	DOVARS	168
25E88	C%ARG	38	25EBA	DPRADIX?	178
25E89	C%ASIN	38	25EBB	DUPGROBDIM	90
25E8A	C%ASINH	38	25EBC	Disp5x7	282
25E8B	C%ATAN	38	25EBD	DispVarsUtil	
25E8C	C%ATANH	38	25EBE	Do1st/2nd+:	278
25E8D	C%COS	38	25EBF	DoBadKey	207
25E8E	C%COSH	38	25EC0	DoCAlarmKey	
25E8F	C%C^C	37	25EC1	DoDelim	302
25E90	C%C^R	37	25EC2	DoDelims	302
25E91	C%EXP	38	25EC3	DoFirstRow	294
25E92	C%LN	38	25EC4	DoHere:	168
25E93	C%LOG	38	25EC5	DoKeyOb	206
25E94	C%R^C	37	25EC6	DoMenuKey	291
25E95	C%SGN	37	25EC7	DoNameKeyLRS	
25E96	C%SIN	38	25EC8	DoNameKeyRS	
25E97	C%SINH	38	25EC9	DoNextRow	
25E98	C%SQRT	37	25ECA	DoPlotMenu	
25E99	C%TAN	38	25ECB	DoPrevRow	
25E9A	C%TANH	38	25ECC	DoSolvrMenu	
25E9B	C>Im%	37	25ECD	DropBadKey	207
25E9C	C>Re%	37	25ECE	EDITDECOMP\$	53
25E9D	CK0ATTNABORT	207	25ECF	EQUATION	318
25E9E	CK1NoBlame	197	25ED0	EVALCRUNCH	
25E9F	CKREF	171	25ED1	Echo2Macros	
25EA0	COERCE\$22	47	25ED2	EditMenu	311
25EA1	CREATEDIR	168	25ED3	EqList?	73
25EA2	CRUNCH	86	25ED4	FlashMsg	282
25EA3	CRUNCHNoBlame		25ED5	GBUFFGROBDIM	273
25EA6	CheckMenuRow	293	25ED6	GETKEY	206
25EA7	Ck&DecKeyLoc	205	25ED7	GETKEY*	206
25EA8	Ck&Freeze		25ED8	GROB>GDISP	91
25EA9	CodePl>%rc.p	205	25ED9	GetKeyOb	206
25EAA	DECOMP\$	53	25EDA	GetMenu%	289
25EAB	DISPROW1*	281	25EDB	GetNextToken	50
25EAC	DISPROW2*	281	25EDC	H/W>KeyCode	205

Addr.	Name	Page	Addr.	Name	Page
25EDD	H/WKey>KeyOb		25F0E	XYGROBDISP	91
25EDE	HARDHEIGHT	273	25F0F	a%>\$	46
25EDF	ImmedEntry?	278	25F0F	a%>\$,	46
25EE0	InitMenu	291	25F10	ederr	158
25EE1	InitMenu%	291	25F11	editdecomp\$w	53
25EE2	InitTrack:	289	25F12	sstDISP	280
25EE3	KEYINBUFFER?	206	25F13	stkdecomp\$w	51
25EE4	KeepUnit	82	25F14	XEQPGDIR	168
25EE5	Key>StdKeyOb	208	25F15	>FONT	283
25EE6	Key>U/SKeyOb	208	25F16	DISP_LINE	
25EE7	LastNonNull	168	25F17	GetMetaVStackDROP	161
25EE8	LoadTouchTbl	292	25F18	GetVStack	161
25EE9	LockAlpha	278	25F19	PopMetaVStack	162
25EEA	ModifierKey?	205	25F1A	PopMetaVStackDROP	160
25EEB	NEXTLIBBAK	99	25F1B	PopVStack	160
25EEC	NULL\$TEMP	46	25F1C	PopVStackAbove	162
25EED	NoAttn?Semi	207	25F1D	PushMetaVStack&Drop	161
25EEE	NoEdit?case	146	25F1E	PushVStack	160
25EEF	NoExitAction	291	25F1F	PushVStack&Clear	160
25EF0	OR\$	50	25F20	PushVStack&Keep	162
25EF1	PATHDIR	169	25F21	PushVStack&KeepDROP	162
25EF2	PrevNonNull	168	25F22	RestoreSysFlags	176
25EF3	RAD?	178	25F23	SaveSysFlags	176
25EF4	RECLAIMDISP	273	25F24	RIGHT\$3x6	96
25EF5	REPEATER		25F25	CKNNOLASTWD	197
25EF6	REPEATERCH		25F29	EvalNoCK:	198
25EF7	SAFE@_HERE	166	25F2A	Keyword?	
25EF8	SEP\$NL	48	25F2B	CHECKMENU	293
25EFA	SLEEPxcp		25F2C	Find1stT.1	
25EFB	SaveLastMenu	289	25F2D	TogInsertKey	
25EFC	SetKeysNS	290	25F63	!REDIMUSER	
25EFD	SetSomeRow	293	25F68	!REDIMTEMP	
25EFE	SetThisRow	292	25F6D	realPAcode	
25EFF	SolvMenuInit	293	25F72	#:>\$	46
25F00	StartMenu	292	25F77	#>\$	46
25F01	Std/BoxLabel	95	25F7C	\$>GROB	96
25F02	StdMenuKeyLS	290	25F81	\$>grob	96
25F03	StdMenuKeyNS	290	25F86	\$>GROBCR	96
25F04	SuspendOK?	224	25F8B	\$>grobCR	96
25F05	TurnOffKey		25F90	>LANGUAGE	179
25F06	UART?		25F95	LANGUAGE>	179
25F07	UARTxcp		25F9A	OLastRomWrd!	197
25F08	UnLockAlpha	278	25F9A	OLASTOWDOB!	197
25F09	UserKeys?	208	25F9F	?ACCPTR>	
25F0A	VLM		25FA4	ABUFF	272
25F0B	WaitForKey	206	25FA9	ALARM?	174
25F0C	XEQStoKey	167	25FAE	ATTN?	207
25F0D	XOR\$	50	25FB3	DISPN	281

25EDD - 26170 563

Addr.	Name	Page	Addr.	Name	Page
25FB3	BIGDISPN	281	26080	GROB!ZERO	91
25FB8	BIGDISPROW1	281	26085	GROBDIM	90
25FB8	DISPROW1	281	2608A	GsstFIN	
25FB8	DISP@01	281	2608F	HARDBUFF	273
25FBD	DISPROW2	281	26094	HARDBUFF2	273
25FBD	DISP@09	281	26099	HEIGHTENGROB	273
25FBD	BIGDISPROW2	281	2609E	INVGROB	91
25FC2	DISP@17	281	260A3	KILLGDISP	273
25FC2	BIGDISPROW3	281	260A8	LastMenuRow!	288
25FC2	DISPROW3	281	260AD	LastMenuRow@	289
25FC7	DISPROW4	281	260B2	MAKEGROB	91
25FC7	BIGDISPROW4	281	260B7	MenuRow!	288
25FC7	DISP@25	281	260BC	MenuRow@	288
25FCC	DISPROW5	281	260C1	NOHALTERR	157
25FD1	DISPROW7	281	260C6	NOTLISTcase	145
25FD6	DISPROW8	281	260CB	NOTROMPcase	145
25FDB	DISPROW9	281	260D0	NOTSECOcase	145
25FE0	DISPROW10	281	260D5	PIXOFF3	91
25FE5	DISPLASTROW	282	260DA	PIXON3	91
25FEA	DISPLASTROWBUT1	282	260DF	PIXOFF	91
25FEF	CENTER\$3x5	96	260E4	PIXON	91
25FF4	CENTER\$5x7	97	260E9	PIXON?3	92
25FF9	LEFT\$3x5	97	260EE	PIXON?	91
25FFE	LEFT\$5x7	97	260F3	PULLCMPEL	64
26003	LEFT\$5x7Arrow	97	260F8	PULLREALEL	64
26008	LEFT\$3x5Arrow	97	260FD	PUTCMPEL	65
2600D	LEFT\$5x7CRArrow	97	26102	PUTEL	65
26012	LEFT\$3x5CRArrow	97	26107	PUTREALEL	65
26017	LEFT\$5x7CR	97	2610C	PrgmEntry?	278
2601C	LEFT\$3x5CR	97	26111	RDUP	129
26021	CLEARVDISP	277	26116	SETCIRCERR	157
2602B	COERCEFLAG	135	2611B	SETCURSOR	
26030	CURSOR_OFF	300	26120	SLOW	172
26035	ClrAlphaAnn	278	26125	VERYSLOW	172
2603A	ClrLeftAnn	277	2612A	VERYVERYSLOW	172
2603F	ClrRightAnn	277	2612F	SUBGROB	91
26044	ClrSysFlag	175	26134	SYNTAXERR	157
26049	ClrUserFlag	175	26139	SetAlphaAnn	278
2604E	DOENG	177	2613E	SetLeftAnn	277
26053	DOFIX	177	26143	SetPrgmEntry	278
26058	DOSCI	177	26148	SetRightAnn	277
2605D	DOSTD	177	2614D	SetSysFlag	175
26062	DropSys0bs		26152	SetUserFlag	175
26067	ERRBEEP	156	26157	SystemLevel?	
2606C	ERASE&LEFT\$5x7	97	26161	TIMEOUT?	
26071	ERASE&LEFT\$3x5	97	26166	TOADISP	272
26076	GBUFF	272	2616B	TOGDISP	272
2607B	GROB!	90	26170	TestSysFlag	175
–					,

Addr.	Name	Page	Addr.	Name	Page
26175	TestUserFlag	175	26260	nDISPSTACK	275
2617F	WINDOWCORNER	279	26265	PushMetaVStack	161
26184	WINDOWDOWN	279	2626A	PutElemBotVStack	163
26189	WINDOWLEFT	279	2626F	PutElemTopVStack	163
2618E	WINDOWRIGHT	279	26274	RCL_NB_AFF_LGN	
26193	WINDOWUP	279	26279	RCL_NB_AFF_LGNSTK	
26198	WINDOWXY	279	2627E	SCANFONT	
2619D	addtics		26283	SetHeader	273
261A2	rstfmt1	178	26288	StackLineHeight	283
261A7	savefmt1	177	2628D	STO_ML_DISP_SIZE	
261AC	setbeep	178	26292	CK0NOLASTWD	196
261B1	stackitw		26297	CK1NOLASTWD	197
261B6	subpdcdptch		2629C	CK2NOLASTWD	197
261BB	tok8trior	50	262A1	CK3NOLASTWD	197
261C0	CLCD10	277	262A6	CK4NOLASTWD	197
261C5	CLEARLCD	277	262AB	CK5NOLASTWD	197
261CA	FLUSHKEYS	205	262B0	CK0	196
261CA	FLUSH	205	262B5	CK1	196
261CF	%%>C%	37	262BA	CK2	196
261D4	C%%0=	38	262BF	CK3	196
261D9	C%%>C%	37	262C4	CK4	196
261DE	C%%CHS	38	262C9	CK5	196
261E3	C%%CONJ	38	262CE	CKN	196
261E8	C%0=	38	262D8	SETSIZEERR	158
261ED	C%CHS	37	262DD	SETTYPEERR	157
261F2	C%CONJ	37	262E2	SETSTACKERR	157
261F7	DISPROW6	281	262E7	SETNONEXTERR	158
261FC	Re>C%	37	262EC	%ABSCOERCE	22
26201	nextsym'R		262F1	COERCE	22
26206	tok8cktrior	50	262F6	UNCOERCE	31
2620B	>MINIFONT	283	262FB	COMPEVAL	128
26210	CHECK_SCAN_FONT		26300	CK1&Dispatch	197
26215	DropVStack	162	26305	CK2&Dispatch	197
2621A	FONT>	283	2630A	CK3&Dispatch	197
2621F	FSCANFONT		2630F	CK4&Dispatch	197
26224	GetElemBotVStack	163	26314	CK5&Dispatch	197
26229	GetElemTopVStack	163	26319	EvalNoCK	198
2622E	SetVStackProtectWord	164	2631E	CK&DISPATCH0	197
26233	GetVStackProtectWord	164	26323	CK&DISPATCH2	197
26238	GetFontCmdHeight		26328	CK&DISPATCH1	197
2623D	GetFontHeight		2632D	#*OVF	23
26242	GetFontStkHeight	283	2639B	MATATLOOP	
26242	StackFontHeight	283	263D2	!MATTRNnc	
26247	GetHeader	273	26459	setStdWid	51
2624C	GetMetaVStack	162	2645E	setStdEditWid	51
26251	InitVirtualStack		2649F	ClrBusyAnn	278
26256	INITMKFONT		264CC	FIRSTC@	300
2625B	MINIFONT>	283	264D1	SETFIRSTC_0	

26175 - 27E9B 565

Addr.	Name	Page	Addr.	Name	Page
264DB	FIRSTC+		27742	#62D	19
2657B	CURSOR_OFF0		2774C	2774C #A01	
26580	CURSOR_OFF+		27756	27756 #A02	
26585	CURSOR@	300	27760	#A04	19
2658A	CURSOR+		2776A	#A05	19
26594	CURSOR_PART	300	27774	#A06	19
265B7	Z>	334	2777E	#12F	17
265BC	Z<	334	27788	#62E	19
265C1	Z=	334	27792	#618	18
265C6	Z<>	334	2779C	#619	18
265CB	Z>=	334	277A6	#61A	18
265D0	Z<=	334	277В0	#61B	18
26F36	<b>%1</b> +	31	277BA	#61C	18
26F4A	%1-	31	277C4	#61D	18
270EE	%1.8	28	277CE	#61E	18
27103	%80	29	277D8	#61F	18
27118	%.1	28	277E2	#620	18
2712D	%.15		277EC	#621	18
27142	LAMLNAME		277F6	#605	18
27155	'IDX	116	27800	#606	18
2716D	StdIOPAR	181	2780A	#607	18
27195	CRLF\$	43	27814	#608	18
271F4	2NULLLAM{}	119	2781E	#609	18
27208	3NULLLAM{}	119	27828	#60A	18
272D9	tok->	44	27832	#60B	18
272F3	ID_EQ	116	2783C	#60C	18
272FE	NULLID	116	27846	#60D	18
275C6	TakeOver	291	27878	#800	19
275EE	Modifier	291	27882	Attn#	19
275FD	MenuKey	293	27937	ID_SIGMADAT	116
27620	MenuMaker	291	27A3A	StdPRTPAR	
2768E	#60E	18	27A89	%%2PI	30
27698	#60F	18	27AA3	NULLGROB	90
276AC	#611	18	27B43	'IDFUNCTION	131
276B6	#612	18	27B6B	'IDPOLAR	131
276C0	#613	18	27B7F	'IDPARAMETER	131
276CA	#614	18	27C33	ExitFcn	
276D4	#615	18	27D3F	CROSSGROB	90
276DE	#616	18	27D5D	MARKGROB	90
276E8	#617	18	27D7B	StdLabelGrob	90
276F2	#622	18	27DBF	C%-1	36
276FC	#623	19	27DE4	C%0	36
27706	#624	19	27E09	C%1	36
27710	#628	19	27E2E	C%%1	36
2771A	#629	19	27E5D	%100	29
27724	#62A	19	27E72	nohalt	224
2772E	#62B	19	27E87	TrueTrue	135
27738	#62C	19	27E9B	failed	136
			1		

Addr.	Name	Page	Addr.	ddr. Name	
27EAF	<skipkey< td=""><td>314</td><td>29ED0</td><td>'Rapndit</td><td>131</td></skipkey<>	314	29ED0	'Rapndit	131
27EFB	>SkipKey	314	29EE9	DaDGNTc	
27F47	<delkey< td=""><td>314</td><td>29F25</td><td colspan="2">25 AppDisplay!</td></delkey<>	314	29F25	25 AppDisplay!	
27F9A	>DelKey	314	29F35	AppDisplay@	223
27FED	NullMenuKey	291	29F55	AppKeys!	223
28001	ROT#1+UNROT	24	29F75	AppKeys0	223
28071	pull	24,77	2A055	AppExitCond!	223
28071	#1-SWAP	24,77	2A065	AppExitCond@	223
28085	pullrev	77	2A085	Clipboard!	307
28099	SWAP#1+SWAP	24	2A095	Clipboard@	307
280AD	SWAP#1-SWAP	24	2A0A5	Clipboard0	307
280C1	ORDERXY#	92	2A0B5	Clipboard?	307
280F8	ORDERXY%	92	2A145	AppError!	223
2812F	SWAPTRUE	136	2A158	AppError@	223
28143	NDUPN	106	2A4FC	WaitTbz0	
28187	reversym	107	2A5CA	SUB\$1#	48
281D5	#1-UNROT	24	2A7A7	CkSecoType	
28211	NDROPFALSE	75, 136	2A7F7	DispTimeReq?	274
28225	9UNROLL	109	2A842	Decomp1Line	52
2825E	2NELCOMPDROP	68	2A85D	DecompEdit	53
282CC	DROP%0	31	2A878	Decomp#Line	52
283E8	FalseFalse	136	2A893	Decomp#Disp	52
283FC	ISTOP-INDEX	152	2A8AE	DecompEcho	54
286E7	symcomp	86	2A8C9	DecompStd1Line	52
28989	SWAPDROP#1-	24	2A8E4	DecompStd1Line32	52
28ACE	DROP?symcomp	86	2A904	RPNDecomp1Line	52
293A3	?symcomp	86	2A924	RPNDecompEdit	53
293F8	P{}N	71	2A944	RPNDecomp#Line	53
2962A	RDROPFALSE	136	2A964	RPNDecomp#Disp	52
2963E	psh&	76	2A984	RPNDecompEcho	54
29693	psh1top&	77	2A9A4	RPNDecompStd1Line	52
296A7	top&	76	2A9C4	RPNDecompStd1Line32	52
2973B	pshtop&	76	2A9E9	RunRPN:	198
29754	pullpsh1&	77	2AA43	AlgDecomp	53
29786	'RSWAP#1+	24	2AA70	CASEVAL	
29821	psh1&	77	2AAE0	SimplifyExpression	
298C0	psh1&rev	77	2AB69	RunInApprox	176
29972	pshzer	77	2ABD7	RunSafeFlagsNoError	177
29986	pshzerpsharg	88	2ABF0	RunSafeFlags	176
29A5D	psh	76	2AC0E	DoRunSafe	177
29A8F	roll2ND	76	2AC72	need'case	
29B12	unroll2ND	76	2ACA9	addrTEMPTOP	
29CB9	uncrunch	87	2ACBO ?TogU/LCase		
29D18	ONE { }N	72	2AD81 EQUALcasedrop		145
29D6A	delimcase		2ADBD		
29E29	ngsizecase		2ADE0 numb1stcase		$\frac{145}{145}$
29E67	nultrior	50			143
29E99	tok=casedrop	145	2AF37		

27EAF - 2C29F 567

Addr.	Name	Page	Addr. Name		Page
2AFFB	MEQ1stcase	143	2BF3A	DA10K?NOTIT	275
2B01B	MEQopscase	143	2BF53	DA2aOK?NOTIT	275
2B06A	AEQopscase	143	2BF6C	DA2bOK?NOTIT	275
2B083	Mid1stcase	143	2BF85	DA3OK?NOTIT	275
2B0CC	idntcase	145	2C039	nWHEREIFTE	
2B0EF	idntlamcase	145	2C044	nWHEREDER	
2B11C	num0=case	142	2C04F	nWHEREINTG	
2B149	%0=case	142	2C05A	nWHERESUM	
2B15D	C%0=case	142	2C065	nWHEREWHERE	
2B176	num1=case	143	2C07B	D/D*	
2B1A3	%1=case	142	2C086	D/D+	
2B1C1	C%1=case	142	2C091	D/D-	
2B1DF	num2=case	143	2C09C	D/D/	
2B20C	%2=case	143	2C0A7	derquot	
2B22A	C%2=case	143	2C0ED	derprod1	
2B25C	num-1=case	143	2C10B	D/D=	
2B289	%-1=case	143	2C116	D/DABS	
2B2A7	C%-1=case	143	2C121	easyabs	
2B2C5	NOTcaseFALSE	140	2C13A	D/DACOS	
2B2F2	CallEditCmd:	311	2C145	D/DACOSH	
2B31A	Roll&Do:		2C150	D/DALOG	
2B351	Rcl&Do:		2C15B	D/DARG	
2B3A6	<pre>1NULLLAM{ }</pre>	119	2C166	D/DASIN	
2B3AB	NULLLAM	116	2C171	D/DASINH	
2B3B7	4NULLLAM{}	119	2C17C	D/DATAN	
2B3FD	%IP>#	31	2C187	D/DATANH	
2B42A	PUTLIST	72	2C192	D/DCHS	
2B475	ParOuterLoop	223	2C1B0	D/DCONJ	
2B4AC	POLSaveUI	223	2C1CE	D/DCOS	
2B542	POLSetUI	223	2C1D9	D/DCOSH	
2B628	POLKeyUI	223	2C1E4	D/DEXP	
2B682	POLErrorTrap		2C1EF	D/DINV	
2B6B4	POLResUI&Err	223	2C1FA	D/DLN	
2B6CD	POLRestoreUI	223	2C205	D/DLNP1	
2B709	InitPOLVars		2C210	D/DLOG	
2B74F	StartupProc		2C21B	D/DIFTE	
2B7CC	addrClkOnNib		2C226	D/DSIN	
2B8BE	OBJ>R	129	2C231	D/DSINH	
2B8E6	R>OBJ	129	2C23C	D/DSQ	
2B90B	'DROPFALSE	130	2C247	D/DSQRT	
2BAB3	COLAthexFCN		2C252	D/DTAN	
2BB21	sscknum2	87	2C25D	D/DTANH	
2BB3A	sncknum2	87	2C268	D/D^	
2BB53	nscknum2	87	2C273	D/D^X	
2BCA2	cknumdsptch1	87	2C27E	E D/D^Y	
2BD8C	SYMBN:	86	2C289		
2BE36	ALGcase	146	2C294	2C294 D/DWHERE	
2BF1C	CkEQUtil		2C29F	D/DINTEGRAL	

Addr.	Name	Page	Addr.	Name	Page
2C2AA	D/DSUM		2E2AA	MakeLabel	293, 96
2C2B5	D/DAPPLY		2EE5A	DispEditLine	274
2C2C0	nCustomMenu	293	2EE5B DispTime?		
2C2CB	nCOLCTQUOTE		2EE5C	BlankDA12	277
2C2D6	SPLITWHERE		2EE5D	?FlashAlert	178
2C2F9	DispStsBound	274	2EE5E	SysErrorTrap	
2C305	DispStatus	274	2EE5F	SysErrorTrapConfirm	
2C311	?DispStatus	274	2EE60	DoWarning	282
2C341	?DispStack	274	2EE61	FlashWarning	282
2C371	DoInputForm	257	2EE62	DA1OK?	275
2C37D	PTYPE>PINFO		2EE63	DA3OK?	275
2C388	MOVEVAR		2EE64	SetDAsTemp	275
2C393	COPYVAR		2EE65	SetDA12a3NoCh	276
2C4D2	#AAA0	20	2EE65	SetDA12a3NCh	276
2D74F	um*	81	2EE66	DA2aLess10K?	275
2D759	um/	81	2EE67	SetDA1Valid	275
2D763	um^	81	2EE68	SetDA2bValid	$\frac{275}{275}$
2D76D	umP	81	2EE69	SetDA1Temp	275
2D777	umEND	81	2EE6A	SetDA2bTemp	275
2D848	tok_q	44	2EE6B	SetDA3Temp	275
2D86D	tok_m	44	2EE6C	SetDA2aEcho	276
2D8AD	tok_s	44	2EE6D	ClrDAsOK	$\frac{275}{275}$
2D0AD 2D949	UMSIGN	83	2EE6E	ClrDA3OK	$\frac{275}{275}$
2D95D	UMIP	83	2EE6F	SetDA12NoCh	276
2D93D 2D971	UMFP	83	2EE01 2EE70	SetDA12NoCh	276
2D971 2D985	UMFLOOR	83	2EE70 2EE71	SetDA13NoCH SetDA12Temp	$\frac{276}{275}$
2D903 2D999	UMCEIL	83	2EE71 2EE72	SetDA121emp SetDA1NoCh	276
2D999 2D9CB	UMRND	83	2EE72 2EE73	SetDA1NOCH	276
2D9CB 2D9EE	UMTRC	83	2EE73	ClrDA1Bad	276
2D9EE 2DA11	cfF	30	2EE74 2EE75	ClrDAlBad ClrDA2aBad	276
	cfC	30	2EE75 2EE76		276
2DA2B		30		SetDA2bNoCh	276
2DCB5	FLOAT		2EE77	SetDA3NoCh	
2DD27	Day>Date		2EE78	SetDA1Bad	276
2DDD5	getBPOFF		2EE79	SetDA2aBad	276
2DE26	mpop1%		2EE7A	SetDA2bBad	276
2DEBB	DAY#	000 074	2EE7B	SetDA3Bad	276
2DFCC	?DispMenu	292, 274	2EE7C	SetDAsNoCh	276
2DFE0	DispMenu	292, 274	2EE7D	ClrDA1IsStat	273
2DFF4	DispMenu.1	292, 274	2EE7E	DA2bIsEdL?	276
2E0D5	Grob>Menu	293, 95	2EE7F SetDA2bIsEdL		276
2E0F3	Str>Menu	293, 95	2EE80 ClrDA2bIsEdL		276
2E107	Seco>Menu	293, 95			276
2E11B	Id>Menu	293, 95			275
2E166	MakeStdLabel	95			
2E189	MakeBoxLabel	95			275
2E1EB	MakeDirLabel	95	2EE8E ClrDA2aOK		275
2E24D	MakeInvLabel	95			275
2E25C	InvLabelGrob	90	2EE90	ClrDA2OK	275

2C2AA - 2EF59 569

Addr.	Name	Page	Addr. Name		Page
2EE91	SetDA2Valid	275	2EED9	2EED9 SYMBNUMSOLVE	
2EE93	SetDA2NoCh	276	2EEDA	STATCLST	66
2EE94	SetDA23NoCh	276	2EEDB	STATSADD%	66
2EEA0	SetDA3ValidF	275	2EEDC	STATN	66
2EEA5	SetDA2bTempF	275	2EEDD	STATSMAX	66
2EEA6	DA2bTemp?		2EEDE	STATMEAN	66
2EEA7	ClrDA2bTemp	275	2EEDF	STATSMIN	66
2EEAB	DA1IsStatus?	276	2EEE0	STATSTDEV	66
2EEAC	SetDAlIsStat	276	2EEE1	STATTOT	66
2EEAE	SetNoRollDA2	276	2EEE2	STATVAR	66
2EEAF	ClrNoRollDA2	276	2EEE3	EchoChrKey	
2EEB0	DA1Bad?	276	2EEE4	Echo\$Key	302
2EEB1	DA2aBad?	276	2EEE5	EditLevel1	311
2EEB2	DA2bBad?	276	2EEE6	InitEd&Modes	314
2EEB3	ClrDA2bBad	276	2EEE7	InitEdLine	303, 314
2EEB4	DA3Bad?	276	2EEE8	${ t InitEdModes}$	314
2EEB5	ClrDA3Bad	276	2EEE9	EditString	310
2EEB7	DA2bNoCh?	276	2EEEA	CURSOR_END?	300
2EEBB	SENDLIST	180	2EEEB	EDITLINE\$	300
2EEBC	GETNAME	180	2EEEC	EDITPARTS	
2EEBD	DOFINISH	180	2EEED	NoEditLine?	300
2EEBE	DOPKT	180	2EEEE	APPprompt1!	
2EEBF	GetIOPAR	181	2EEEF	AUTOSCALE	318
2EEC0	DOOPENIO		2EEF0	PromptIdUtil	48
2EEC1	DOBAUD	180	2EEF1	PUTSCALE	318
2EEC2	DOPARITY	180	2EEF2	PUTINDEP	317
2EEC3	DOTRANSIO	180	2EEF3	PUTINDEPLIST	317
2EEC4	DOKERRM	180	2EEF4	PUTRES	317
2EEC5	DOBUFLEN	180	2EEF5	GETPTYPE	317
2EEC6	DOSBRK	180	2EEF6	PUTPTYPE	317
2EEC7	DOSRECV	180	2EEF7	VSCALE	
2EEC8	FLUSHRSBUF		2EEF8	HSCALE	
2EEC9	CLOSEUART	180	2EEF9	DOERASE	273
2EECA	docr		2EEFA	CROSS_HAIRS	
2EECB	DOCR	181	2EEFB	CROSS_OFF	
2EECC	DOPRLCD		2EEFC	MENUOFF	273
2EECD	DODELAY	181	2EEFD	MENUOFF?	273
2EECE	SetEcma94		2EEFE	CURRENTMARK?	
2EECF	TOD	172	2EEFF	DispCoord1	$\frac{282}{318}$
2EED0	DATE	173	2EF01	DOPX>C	
2EED1	DDAYS	173	2EF02	DOC>PX	
2EED2	DATE+DAYS	173	2EF03	DOLCD>	
2EED3	TIMESTR	49, 173	2EF04		
2EED4	Clr8	277	2EF05		
2EED5	Clr8-15	277	2EF06		
2EED6	HBUFF_X_Y	279	2EF07 nmetasyms		199
2EED7	SysTime	173	2EF26	SYMSHOW	88
2EED7	CLKTICKS	173	2EF59	MENP&FixDA12	

ZEF5A	Addr.	Name	Page	Addr. Name		Page
2EF5F	2EF5A	apndvarlst	73	2EF91	CAL_CURS_POS	301
DOGRAPHIC   318   ZEF94	2EF5E	BlankDA1	277	2EF92	XLINE_SIZE?	314
28F61	2EF5F	InputLine	212	2EF93	VERIF_SELECTION	306
Def   Def	2EF60	DOGRAPHIC	318	2EF94	PASTE.EXT	307
2EF66	2EF61	WINDOW#	281	2EF95	DEL_CMD	303
2EF67	2EF62	palparse	50	2EF96	NO_AFFCMD	312
ZEF68	2EF66	SysMenuCheck	292	2EF97	InsertEcho	302
2EF69	2EF67	SysDisplay	274	2EF98	SetDA2aValid	275
ZEF6A   Parse.1   ZEF9F	2EF68	ClrDouseAlm		2EF99	SetDA3Valid	275
ZEF6B         Parse.2         ZEFAO         LINEOFF         92           ZEF6C         AtUserStack         197         ZEFA1         TOGLINE         92           ZEF6D         GetLastEdit         2EFA2         LINEON3         92           ZEF6E         ParseFail         51         ZEFA3         LINEOFF3         92           ZEF7D         DispBadToken         51         ZEFA5         DOHEX         177           ZEF71         DispBadToken2         ZEFA6         DOBIN         177           ZEF72         CacheStack         ZEFA6         DOBIN         177           ZEF73         POOCT         177         177           ZEF74         CMD_PLUS         301         ZEFAA         DOBEC         177           ZEF73         PSpace/Go>         311         ZEFAA         DOCT         177           ZEF73         PSpace/Go>         311         ZEFAA         DOBEC         177           ZEF74         CMD_PLUS         301         ZEFAA         DOBEC         177           ZEF75         AddTrailingSpace         312         ZEFAA         Dotton         58           ZEF77         CMD_PAGER         305         ZEFAE         bit	2EF69	EvalParsed		2EF9A	CommandLineHeight	312
2EF6C	2EF6A	Parse.1		2EF9F	LINEON	92
2EF6D   GetLastEdit   2EFA2	2EF6B	Parse.2		2EFA0	LINEOFF	92
2EF6E         ParseFail         51         2EFA3         LINEOFF3         92           2EF70         ParseFail2         2EFA5         DOHEX         177           2EF71         DispBadToken2         2EFA6         DOBIN         177           2EF72         CacheStack         2EFA7         DOOCT         177           2EF73         ?Space/Go>         311         2EFA8         DODEC         177           2EF74         CMD_PLUS         301         2EFAA         dostws         57           2EF75         AddTrailingSpace         312         2EFAC         bitAND         58           2EF76         AddLeadingSpace         312         2EFAD         bitOR         58           2EF77         CMD_PAGEL         305         2EFAE         bitNOT         58           2EF77         CMD_PAGER         305         2EFAE         bitNOT         58           2EF78         CMD_PAGED         305         2EFAB         bitNOT         58           2EF77         CMD_PAGED         305         2EFBB         bitSL         58           2EF78         CMD_PAGED         305         2EFBB         bitSR         58           2EF7B         CMD_	2EF6C	AtUserStack	197	2EFA1	TOGLINE	92
2EF6E         ParseFail         51         2EFA3         LINEOFF3         92           2EF70         ParseFail2         2EFA5         DOHEX         177           2EF71         DispBadToken2         2EFA6         DOBIN         177           2EF72         CacheStack         2EFA7         DOOCT         177           2EF73         ?Space/Go>         311         2EFA8         DODEC         177           2EF74         CMD_PLUS         301         2EFAA         dostws         57           2EF75         AddTrailingSpace         312         2EFAC         bitAND         58           2EF76         AddLeadingSpace         312         2EFAD         bitOR         58           2EF77         CMD_PAGEL         305         2EFAE         bitNOT         58           2EF77         CMD_PAGER         305         2EFAE         bitNOT         58           2EF78         CMD_PAGED         305         2EFAB         bitNOT         58           2EF77         CMD_PAGED         305         2EFBB         bitSL         58           2EF78         CMD_PAGED         305         2EFBB         bitSR         58           2EF7B         CMD_	2EF6D	GetLastEdit		2EFA2	LINEON3	92
2EF70         ParseFail2         2EFA5         DOHEX         177           2EF71         DispBadToken2         2EFA6         DOBIN         177           2EF72         CacheStack         2EFA7         DOOCT         177           2EF73         ?Space/Go>         311         2EFA8         DODEC         177           2EF74         CMD_PLUS         301         2EFAB         DODEC         177           2EF75         AddTrailingSpace         312         2EFAD         DitAND         58           2EF76         AddLeadingSpace         312         2EFAD         DitOR         58           2EF77         CMD_PAGEL         305         2EFAE         DitXOR         58           2EF77         CMD_PAGER         305         2EFAB         DitNOT         58           2EF78         CMD_PAGED         305         2EFB0         DitSL         58           2EF79         CMD_PAGED         305         2EFB1         DitSLB         58           2EF70         CMD_PAGED         305         2EFB1         DitSLB         58           2EF70         CMD_DAK         305         2EFB1         DitSLB         58           2EF70         CMD_DEB	2EF6E	ParseFail	51	2EFA3	LINEOFF3	92
DispBadToken2	2EF6F	DispBadToken	51	2EFA4	TOGLINE3	92
2EF72         CacheStack         2EFA7         DOOCT         177           2EF73         ?Space/Go>         311         2EFA8         DODEC         177           2EF74         CMD_PLUS         301         2EFAA         dostws         57           2EF75         AddTrailingSpace         312         2EFAC         bitAND         58           2EF76         AddLeadingSpace         312         2EFAD         bitCNG         58           2EF77         CMD_PAGEL         305         2EFAE         bitNOT         58           2EF78         CMD_PAGER         305         2EFB0         bitSL         58           2EF79         CMD_PAGEU         305         2EFB0         bitSL         58           2EF70         CMD_PAGED         305         2EFB1         bitSLB         58           2EF70         CMD_BAK         305         2EFB2         bitSR         58           2EF70         CMD_DEB_LINE         305         2EFB3         bitRRB         58           2EF71         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF71         CMD_DEB_LINE         305         2EFB5         bitRRB         58 <tr< td=""><td>2EF70</td><td>ParseFail2</td><td></td><td>2EFA5</td><td>DOHEX</td><td>177</td></tr<>	2EF70	ParseFail2		2EFA5	DOHEX	177
2EF72	2EF71	DispBadToken2		2EFA6	DOBIN	177
2EF74         CMD_PLUS         301         2EFAA         dostws         57           2EF75         AddTrailingSpace         312         2EFAD         bitAND         58           2EF76         AddLeadingSpace         312         2EFAD         bitOR         58           2EF77         CMD_PAGEL         305         2EFAE         bitXOR         58           2EF78         CMD_PAGER         305         2EFAD         bitNOT         58           2EF79         CMD_PAGED         305         2EFBD         bitSL         58           2EF79         CMD_PAGED         305         2EFB1         bitSLB         58           2EF78         CMD_BAK         305         2EFB2         bitSR         58           2EF70         CMD_DAKT         306         2EFB3         bitSRB         58           2EF70         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF77         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF77         CMD_DED_LINE         305         2EFB4         bitRR         58           2EF76         CMD_DUP         305         2EFB5         bitRL         58	2EF72	=		2EFA7	DOOCT	177
2EF74         CMD_PLUS         301         2EFAA         dostws         57           2EF75         AddTrailingSpace         312         2EFAC         bitAND         58           2EF76         AddLeadingSpace         312         2EFAD         bitOR         58           2EF77         CMD_PAGEL         305         2EFAE         bitXOR         58           2EF78         CMD_PAGER         305         2EFAD         bitNOT         58           2EF79         CMD_PAGED         305         2EFBAD         bitNOT         58           2EF79         CMD_PAGED         305         2EFBD         bitSL         58           2EF79         CMD_PAGED         305         2EFB1         bitSLB         58           2EF78         CMD_BAK         305         2EFB2         bitSR         58           2EF70         CMD_DAKT         305         2EFB3         bitSRB         58           2EF70         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF71         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF76         CMD_DUP         305         2EFB4         bitRR         58     <	2EF73	?Space/Go>	311	2EFA8	DODEC	
2EF75         AddTrailingSpace         312         2EFAC         bitAND         58           2EF76         AddLeadingSpace         312         2EFAD         bitOR         58           2EF77         CMD_PAGEL         305         2EFAE         bitXOR         58           2EF78         CMD_PAGER         305         2EFAE         bitXOR         58           2EF79         CMD_PAGEU         305         2EFBO         bitNOT         58           2EF74         CMD_PAGED         305         2EFBO         bitSL         58           2EF7A         CMD_PAGED         305         2EFB1         bitSLB         58           2EF7A         CMD_BAK         305         2EFB2         bitSR         58           2EF7D         CMD_DAT         305         2EFB3         bitSRB         58           2EF7D         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF7E         CMD_DEB_LINE         305         2EFB5         bitRRB         58           2EF7E         CMD_DEB_LINE         305         2EFB5         bitRRB         58           2EF7E         CMD_DUP         305         2EFB5         bitRL         58	2EF74	-	301	2EFAA	dostws	57
2EF76         AddLeadingSpace         312         2EFAD         bitOR         58           2EF77         CMD_PAGEL         305         2EFAE         bitXOR         58           2EF78         CMD_PAGER         305         2EFAF         bitNOT         58           2EF79         CMD_PAGEU         305         2EFBO         bitSL         58           2EF7A         CMD_PAGED         305         2EFB1         bitSLB         58           2EF7B         CMD_PAGED         305         2EFB2         bitSLB         58           2EF7B         CMD_BAK         305         2EFB2         bitSRB         58           2EF7C         CMD_NXT         305         2EFB3         bitSRB         58           2EF7C         CMD_NXT         305         2EFB3         bitRRB         58           2EF7C         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF7E         CMD_DED_LINE         305         2EFB5         bitRRB         58           2EF7E         CMD_UP         305         2EFB5         bitRLB         58           2EF81         CMD_DOWN         305         2EFB6         bitASR         58	2EF75	AddTrailingSpace	312	2EFAC	bitAND	58
2EF77         CMD_PAGEL         305         2EFAE         bitXOR         58           2EF78         CMD_PAGER         305         2EFAF         bitNOT         58           2EF79         CMD_PAGEU         305         2EFB0         bitSL         58           2EF7A         CMD_PAGED         305         2EFB1         bitSLB         58           2EF7B         CMD_BAK         305         2EFB2         bitSR         58           2EF7C         CMD_NXT         305         2EFB3         bitSRB         58           2EF7D         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF7E         CMD_END_LINE         305         2EFB5         bitRRB         58           2EF7E         CMD_END_LINE         305         2EFB6         bitRL         58           2EF7E         CMD_END_LINE         305         2EFB5         bitRRB         58	2EF76			2EFAD	bitOR	
2EF78         CMD_PAGER         305         2EFAF         bitNOT         58           2EF79         CMD_PAGEU         305         2EFB0         bitSL         58           2EF7A         CMD_PAGED         305         2EFB1         bitSLB         58           2EF7B         CMD_BAK         305         2EFB2         bitSR         58           2EF7C         CMD_NXT         305         2EFB3         bitSRB         58           2EF7D         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF7E         CMD_END_LINE         305         2EFB5         bitRRB         58           2EF7F         CMD_END_LINE         305         2EFB6         bitRR         58           2EF7F         CMD_END_LINE         305         2EFB6         bitRR         58           2EF7F         CMD_END_LINE         305         2EFB6         bitRR         58           2EF7F         CMD_END_LINE         305         2EFB6         bitRL         58           2EF7F         CMD_UP         305         2EFB6         bitRL         58           2EF80         CMD_DOWN         305         2EFB7         bit+         57	2EF77		305	2EFAE	bitXOR	58
2EF79         CMD_PAGEU         305         2EFB0         bitSL         58           2EF7A         CMD_PAGED         305         2EFB1         bitSLB         58           2EF7B         CMD_BAK         305         2EFB2         bitSR         58           2EF7C         CMD_NXT         305         2EFB3         bitSRB         58           2EF7D         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF7E         CMD_END_LINE         305         2EFB5         bitRRB         58           2EF7E         CMD_END_LINE         305         2EFB6         bitRRB         58           2EF7E         CMD_DUP         305         2EFB6         bitRLB         58           2EF7E         CMD_UP         305         2EFB6         bitRL         58           2EF80         CMD_DOWN         305         2EFB6         bitRLB         58           2EF81         CMD_DROP         302         2EFB8         bitASR         58           2EF82         CMD_DEDL         302         2EFB8         bit+         57           2EF83         CMD_STO_FIN         306         2EFBC         bit/         57		<del>_</del>	305		bitNOT	
2EF7A         CMD_PAGED         305         2EFB1         bitSLB         58           2EF7B         CMD_BAK         305         2EFB2         bitSR         58           2EF7C         CMD_NXT         305         2EFB3         bitSRB         58           2EF7D         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF7E         CMD_END_LINE         305         2EFB5         bitRRB         58           2EF7E         CMD_DUP         305         2EFB6         bitRL         58           2EF7F         CMD_UP         305         2EFB6         bitRL         58           2EF80         CMD_DOWN         305         2EFB6         bitRL         58           2EF81         CMD_DOOP         302         2EFB8         bitASR         58           2EF82         CMD_DED         302         2EFB8         bit+         57           2EF83         CMD_STO_DEBUT         306         2EFBA         bit-         57           2EF84         CMD_STO_FIN         306         2EFBD         bit/         57           2EF86         RCL_CMD_FIN         306         2EFBE         WORDSIZE         57			305	2EFB0	bitSL	
2EF7B         CMD_BAK         305         2EFB2         bitSR         58           2EF7C         CMD_NXT         305         2EFB3         bitSRB         58           2EF7D         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF7E         CMD_END_LINE         305         2EFB5         bitRRB         58           2EF7F         CMD_UP         305         2EFB6         bitRL         58           2EF80         CMD_DOWN         305         2EFB7         bitRLB         58           2EF81         CMD_DROP         302         2EFB8         bitASR         58           2EF82         CMD_DROP         302         2EFB8         bitASR         58           2EF83         CMD_DROP         302         2EFB8         bit+ASR         58           2EF84         CMD_DEL         302         2EFB9         bit+         57           2EF83         CMD_STO_DEBUT         306         2EFBA         bit-         57           2EF84         CMD_STO_FIN         306         2EFBC         bit/         57           2EF85         RCL_CMD_POB         306         2EFBB         WORDSIZE         57		<del>-</del>	305		bitSLB	58
2EF7C         CMD_NXT         305         2EFB3         bitSRB         58           2EF7D         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF7E         CMD_END_LINE         305         2EFB5         bitRRB         58           2EF7F         CMD_UP         305         2EFB6         bitRL         58           2EF80         CMD_DOWN         305         2EFB7         bitRLB         58           2EF81         CMD_DOPO         302         2EFB8         bitASR         58           2EF82         CMD_DEL         302         2EFB9         bit+         57           2EF83         CMD_STO_DEBUT         306         2EFBA         bit-         57           2EF84         CMD_STO_FIN         306         2EFBC         bit*         57           2EF85         RCL_CMD_DEB         306         2EFBC         bit/         57           2EF86         RCL_CMD_FIN         306         2EFBE         WORDSIZE         57           2EF87         RCL_CMD_POS         300         2EFBF         BASE         177           2EF88         CMD_CUT         306         2EFC0         HXS>\$         46 <t< td=""><td></td><td></td><td>305</td><td>2EFB2</td><td></td><td></td></t<>			305	2EFB2		
2EF7D         CMD_DEB_LINE         305         2EFB4         bitRR         58           2EF7E         CMD_END_LINE         305         2EFB5         bitRRB         58           2EF7F         CMD_UP         305         2EFB6         bitRL         58           2EF80         CMD_DOWN         305         2EFB7         bitRLB         58           2EF81         CMD_DOWN         302         2EFB8         bitASR         58           2EF82         CMD_DROP         302         2EFB8         bitASR         58           2EF82         CMD_DEL         302         2EFB9         bit+         57           2EF83         CMD_STO_DEBUT         306         2EFBA         bit-         57           2EF84         CMD_STO_FIN         306         2EFBC         bit*         57           2EF85         RCL_CMD_DEB         306         2EFBD         bit/         57           2EF86         RCL_CMD_FIN         306         2EFBE         WORDSIZE         57           2EF87         RCL_CMD_POS         300         2EFBF         BASE         177           2EF88         CMD_CUT         306         2EFC0         HXS>\$         46      <	2EF7C			2EFB3	bitSRB	
2EF7E         CMD_END_LINE         305         2EFB5         bitRRB         58           2EF7F         CMD_UP         305         2EFB6         bitRL         58           2EF80         CMD_DOWN         305         2EFB7         bitRLB         58           2EF81         CMD_DOOP         302         2EFB8         bitASR         58           2EF82         CMD_DEL         302         2EFB9         bit+         57           2EF83         CMD_STO_DEBUT         306         2EFBA         bit-         57           2EF84         CMD_STO_DEBUT         306         2EFBC         bit*         57           2EF84         CMD_STO_FIN         306         2EFBC         bit/         57           2EF85         RCL_CMD_DEB         306         2EFBD         bit/         57           2EF86         RCL_CMD_FIN         306         2EFBE         WORDSIZE         57           2EF87         RCL_CMD_POS         300         2EFBF         BASE         177           2EF88         CMD_CUT         306         2EFC0         HXS>\$         46           2EF8B         STO_CURS_POS         304         2EFC2         bit##/         57		<del>-</del>		2EFB4	bitRR	
2EF7F         CMD_UP         305         2EFB6         bitRL         58           2EF80         CMD_DOWN         305         2EFB7         bitRLB         58           2EF81         CMD_DROP         302         2EFB8         bitASR         58           2EF82         CMD_DEL         302         2EFB9         bit+         57           2EF83         CMD_STO_DEBUT         306         2EFBA         bit-         57           2EF84         CMD_STO_FIN         306         2EFBC         bit*         57           2EF85         RCL_CMD_DEB         306         2EFBD         bit/         57           2EF86         RCL_CMD_DEB         306         2EFBE         WORDSIZE         57           2EF87         RCL_CMD_POS         300         2EFBF         BASE         177           2EF88         CMD_CUT         306         2EFC0         HXS>\$         46           2EF88         STO_CURS_POS         304         2EFC2         bit%#/         57           2EF8C         STO_CURS_POS3         304         2EFC3         bit%#*         57           2EF8E         STO_CURS_POS_VIS         304         2EFC5         bit%#-         57     <	2EF7E		305	2EFB5	bitRRB	
2EF80         CMD_DOWN         305         2EFB7         bitRLB         58           2EF81         CMD_DROP         302         2EFB8         bitASR         58           2EF82         CMD_DEL         302         2EFB9         bit+         57           2EF83         CMD_STO_DEBUT         306         2EFBA         bit-         57           2EF84         CMD_STO_FIN         306         2EFBC         bit*         57           2EF85         RCL_CMD_DEB         306         2EFBD         bit/         57           2EF86         RCL_CMD_FIN         306         2EFBE         WORDSIZE         57           2EF87         RCL_CMD_POS         300         2EFBF         BASE         177           2EF88         CMD_CUT         306         2EFC0         HXS>\$         46           2EF8A         CMD_COPY         306         2EFC1         hxs>\$         46           2EF8B         STO_CURS_POS         304         2EFC2         bit##/         57           2EF8C         STO_CURS_POS3         304         2EFC4         bit##*         57           2EF8E         STO_CURS_POS_VIS         304         2EFC6         bit##-         57	2EF7F				bitRL	
2EF81         CMD_DROP         302         2EFB8         bitASR         58           2EF82         CMD_DEL         302         2EFB9         bit+         57           2EF83         CMD_STO_DEBUT         306         2EFBA         bit-         57           2EF84         CMD_STO_FIN         306         2EFBC         bit*         57           2EF85         RCL_CMD_DEB         306         2EFBD         bit/         57           2EF86         RCL_CMD_FIN         306         2EFBE         WORDSIZE         57           2EF87         RCL_CMD_POS         300         2EFBF         BASE         177           2EF88         CMD_CUT         306         2EFC0         HXS>\$         46           2EF8A         CMD_COPY         306         2EFC1         hxs>\$         46           2EF8B         STO_CURS_POS         304         2EFC2         bit##/         57           2EF8C         STO_CURS_POS3         304         2EFC4         bit#*         57           2EF8E         STO_CURS_POS_VIS         304         2EFC5         bit#*         57           2EF8F         STO_CURS_POS_VIS         304         2EFC6         bit##         57 <td>2EF80</td> <td></td> <td>305</td> <td>2EFB7</td> <td>bitRLB</td> <td></td>	2EF80		305	2EFB7	bitRLB	
2EF82         CMD_DEL         302         2EFB9         bit+         57           2EF83         CMD_STO_DEBUT         306         2EFBA         bit-         57           2EF84         CMD_STO_FIN         306         2EFBC         bit*         57           2EF85         RCL_CMD_DEB         306         2EFBD         bit/         57           2EF86         RCL_CMD_FIN         306         2EFBE         WORDSIZE         57           2EF87         RCL_CMD_POS         300         2EFBF         BASE         177           2EF88         CMD_CUT         306         2EFC0         HXS>\$         46           2EF8A         CMD_COPY         306         2EFC1         hxs>\$         46           2EF8B         STO_CURS_POS         304         2EFC2         bit*#/         57           2EF8C         STO_CURS_POS3         304         2EFC4         bit*#*         57           2EF8E         STO_CURS_POS_VIS         304         2EFC5         bit***         57           2EF8F         STO_CURS_POS_VIS         304         2EFC6         bit*#-         57		<del>-</del>	302	2EFB8	bitASR	
2EF83         CMD_STO_DEBUT         306         2EFBA         bit-         57           2EF84         CMD_STO_FIN         306         2EFBC         bit*         57           2EF85         RCL_CMD_DEB         306         2EFBD         bit/         57           2EF86         RCL_CMD_FIN         306         2EFBE         WORDSIZE         57           2EF87         RCL_CMD_POS         300         2EFBF         BASE         177           2EF88         CMD_CUT         306         2EFC0         HXS>\$         46           2EF8A         CMD_COPY         306         2EFC1         hxs>\$         46           2EF8B         STO_CURS_POS         304         2EFC2         bit*#/         57           2EF8C         STO_CURS_POS3         304         2EFC4         bit*#*         57           2EF8E         STO_CURS_POS4         304         2EFC5         bit***         57           2EF8F         STO_CURS_POS_VIS         304         2EFC6         bit*#-         57		<del>-</del>			bit+	
2EF84         CMD_STO_FIN         306         2EFBC         bit*         57           2EF85         RCL_CMD_DEB         306         2EFBD         bit/         57           2EF86         RCL_CMD_FIN         306         2EFBE         WORDSIZE         57           2EF87         RCL_CMD_POS         300         2EFBF         BASE         177           2EF88         CMD_CUT         306         2EFC0         HXS>\$         46           2EF8A         CMD_COPY         306         2EFC1         hxs>\$         46           2EF8B         STO_CURS_POS         304         2EFC2         bit*#/         57           2EF8C         STO_CURS_POS3         304         2EFC3         bit*#*         57           2EF8E         STO_CURS_POS4         304         2EFC5         bit***         57           2EF8F         STO_CURS_POS_VIS         304         2EFC6         bit*#-         57		<del>-</del>			bit-	
2EF85         RCL_CMD_DEB         306         2EFBD         bit/         57           2EF86         RCL_CMD_FIN         306         2EFBE         WORDSIZE         57           2EF87         RCL_CMD_POS         300         2EFBF         BASE         177           2EF88         CMD_CUT         306         2EFC0         HXS>\$         46           2EF8A         CMD_COPY         306         2EFC1         hxs>\$         46           2EF8B         STO_CURS_POS         304         2EFC2         bit%#/         57           2EF8C         STO_CURS_POS3         304         2EFC3         bit%#*         57           2EF8E         STO_CURS_POS4         304         2EFC5         bit#**         57           2EF8F         STO_CURS_POS_VIS         304         2EFC6         bit%#-         57			306		bit*	
2EF86         RCL_CMD_FIN         306         2EFBE         WORDSIZE         57           2EF87         RCL_CMD_POS         300         2EFBF         BASE         177           2EF88         CMD_CUT         306         2EFC0         HXS>\$         46           2EF8A         CMD_COPY         306         2EFC1         hxs>\$         46           2EF8B         STO_CURS_POS         304         2EFC2         bit%#/         57           2EF8C         STO_CURS_POS2         304         2EFC3         bit#%*/         58           2EF8D         STO_CURS_POS3         304         2EFC4         bit%#*         57           2EF8E         STO_CURS_POS4         304         2EFC5         bit#**         57           2EF8F         STO_CURS_POS_VIS         304         2EFC6         bit%#-         57	2EF85			2EFBD	bit/	
2EF87         RCL_CMD_POS         300         2EFBF         BASE         177           2EF88         CMD_CUT         306         2EFC0         HXS>\$         46           2EF8A         CMD_COPY         306         2EFC1         hxs>\$         46           2EF8B         STO_CURS_POS         304         2EFC2         bit%#/         57           2EF8C         STO_CURS_POS2         304         2EFC3         bit#%*/         58           2EF8D         STO_CURS_POS3         304         2EFC4         bit%#*         57           2EF8E         STO_CURS_POS4         304         2EFC5         bit#**         57           2EF8F         STO_CURS_POS_VIS         304         2EFC6         bit%#-         57	2EF86		306	2EFBE		57
2EF88         CMD_CUT         306         2EFC0         HXS>\$         46           2EF8A         CMD_COPY         306         2EFC1         hxs>\$         46           2EF8B         STO_CURS_POS         304         2EFC2         bit%#/         57           2EF8C         STO_CURS_POS2         304         2EFC3         bit#%/         58           2EF8D         STO_CURS_POS3         304         2EFC4         bit%#*         57           2EF8E         STO_CURS_POS4         304         2EFC5         bit#%*         57           2EF8F         STO_CURS_POS_VIS         304         2EFC6         bit%#-         57			300	2EFBF		177
2EF8A         CMD_COPY         306         2EFC1         hxs>\$         46           2EF8B         STO_CURS_POS         304         2EFC2         bit%#/         57           2EF8C         STO_CURS_POS2         304         2EFC3         bit#%/         58           2EF8D         STO_CURS_POS3         304         2EFC4         bit%#*         57           2EF8E         STO_CURS_POS4         304         2EFC5         bit#**         57           2EF8F         STO_CURS_POS_VIS         304         2EFC6         bit%#-         57						
2EF8B         STO_CURS_POS         304         2EFC2         bit%#/         57           2EF8C         STO_CURS_POS2         304         2EFC3         bit#%/         58           2EF8D         STO_CURS_POS3         304         2EFC4         bit%#*         57           2EF8E         STO_CURS_POS4         304         2EFC5         bit#**         57           2EF8F         STO_CURS_POS_VIS         304         2EFC6         bit%#-         57						
2EF8C       STO_CURS_POS2       304       2EFC3       bit#%/       58         2EF8D       STO_CURS_POS3       304       2EFC4       bit%#*       57         2EF8E       STO_CURS_POS4       304       2EFC5       bit#%*       57         2EF8F       STO_CURS_POS_VIS       304       2EFC6       bit%#-       57						
2EF8D       STO_CURS_POS3       304       2EFC4       bit%#*       57         2EF8E       STO_CURS_POS4       304       2EFC5       bit#**       57         2EF8F       STO_CURS_POS_VIS       304       2EFC6       bit%#-       57						
2EF8E         STO_CURS_POS4         304         2EFC5         bit#%*         57           2EF8F         STO_CURS_POS_VIS         304         2EFC6         bit%#-         57						
2EF8F STO_CURS_POS_VIS 304 2EFC6 bit%#- 57		<del>-</del>				
		<del>-</del>				
	2EF90	CAL_CURS_POS_VIS	301	2EFC7	bit#%-	57

2EF5A - 2F162 571

2EFC8	Addr.	Name	Page	Addr. Name		Page
ZEFCB	2EFC8	bit%#+	57	2F091	2F091 UMSIN	
Septical Number   Septical N	2EFC9	bit#%+	57	2F092	2F092 UMSQ	
ZEFCC	2EFCA	HXS>%	31	2F093	2F093 UMSQRT	
ZEPCD	2EFCB	%>#	56	2F094	UMTAN	83
Defice	2EFCC	HXS==HXS	58	2F095	UMU>	82
Deficit	2EFCD	HXS>HXS	59	2F096	UMXROOT	83
ZEFDB	2EFCE	HXS>=HXS	59	2F098	Unbr>U	82
2FEC	2EFCF	HXS <hxs< td=""><td>59</td><td>2F099</td><td>U&gt;NCQ</td><td>82</td></hxs<>	59	2F099	U>NCQ	82
Ticks	2EFDB	GROB+	90	2F09A	TempConv	82
Ticks   Tick	2EFEC	symbn		2F0A1	RESETDEPTH	107
Ticks>Rpt	2F002	Ticks>TOD	173	2F0AC	PURGALARM%	174
2F007	2F003	Ticks>Date	173	2F0BC	PRINT	
2F008   getypos   2F005   NEWINDEP   2F019   UNIT>\$   82   2F00B   MAKEPICT#   91   2F031   TURNMENUON   273   2F066   KERMOPEN   2F034   TURNMENUOFF   273   2F067   InitioEnv   2F05E   SaveLastedit   314   2F068   INDEPVAR   317   2F062   StoIOPAR   181   2F06C   ICMPDRPRTDRP   71   2F063   StoPRTPAR   2F06E   HXS#HXS   58   2F066   ST0APPLDATA   2F06F   GETXMAX   316   2F073   SWAPcompSWAP   86   2F06F   GETXMIN   316   2F075   InitSySUI   2F100   GETXMIN   317   2F076   puretemp?   84   2F105   GDISPCENTER   318   2F077   UM>U   82   2F106   GETINDEP   317   2F078   U>nbr   82   2F107   GETPMIN&MAX   317   2F076   UM*?   83   2F108   GETXHS   2F077   UM*?   83   2F108   GETXHS   317   2F076   UM**   83   2F109   GETXPOS   2F07E   UM*CH   83   2F109   GETXPOS   318   2F077   UM**   83   2F109   GETXPOS   317   2F078   UM**   83   2F100   GETXPAS   317   2F078   UM**   83   2F100   GETXPAS   317   2F076   UM**   83   2F100   GETXPAS   317   2F078   UM**   83   2F100   GETXPAS   317   2F080   UM**   83   2F100   GETXPAS   86   2F082   UM-   83   2F110   FINDVARS   86   2F082   UM-   83   2F110   FINDVARS   86   2F082   UM-   83   2F110   Echo\$Nochr00   302   2F085   UM<*   83   2F110   CK#-   23   2F086   UM   83   2F135   CKADU**   2F088   UM ?   83   2F135   CKADU**   2F088   UM>?   83   2F155   input   3   2F122   2F088   UMCHS   83   2F155   input   3   2F122   2F08C   UMCONV   82   2F158   CHADU**   2F08E   UMMAX   83   2F156   CLEARMENU   293   2F08F   UMMIN   2F08F   UMMIN   2F08F   UMMIN   2F08F   UMMIN   2F156   CLEARMENU   2F08F   UMMIN   2F08F	2F004	Ticks>Rpt	173	2F0C5	PLOTPREP	
2F019	2F007	getxpos		2F0D4	ILnot?	199
TURNMENUON   273   2F0E6   KERMOPEN   2F034   TURNMENUOFF   273   2F0E7   InitIOEnv   2F05E   SaveLastEdit   314   2F0E8   INDEPVAR   317   2F062   StoIOPAR   181   2F0EC   ICMPDRPTDRP   71   2F063   StoPRTPAR   2F0EE   HXS#HXS   58   2F064   Sys@   166   2F0EF   HXS<+HXS   59   2F066   STOAPPLDATA   2F0FE   GETXMAX   316   2F073   SWAPCOMPSWAP   86   2F0FF   GETXMIN   317   2F075   InitSysUI   2F100   GETYMIN   317   2F076   puretemp?   84   2F105   GDISPCENTER   318   2F077   UM>U   82   2F106   GETINDEP   317   2F078   U>nbr   82   2F107   GETMIN&MAX   317   2F07E   UM*?   83   2F108   GETRES   318   2F107   UM**   83   2F108   GETRES   317   2F07E   UM*T   83   2F100   GETYMAX   317   2F07E   UM*T   83   2F100   GETRES   317   2F080   UM*   83   2F100   GETYMAX   317   2F081   UM+   83   2F100   GETYMAX   317   2F081   UM+   83   2F100   GETYMAX   317   2F081   UM+   83   2F110   GETYMAX   317   2F081   UM+   83   2F110   GETYMAX   317   2F081   UM+   83   2F110   FINDVARS   86   2F082   UM-   83   2F135   DOOIdMatrix   2F085   UM<=?   83   2F135   DOOIdMatrix   2F086   UM   83   2F135   DRAWLINE#3   92   2F088   UM ?   83   2F155   Input   3   2F12   2F08B   UMCHS   83   2F155   Input   3   2F12   2F08C   UMCONV   82   2F158   THISCHAR   300   2F08E   UMMAX   83   2F15E   CLEARMENU   293   2F08F   UMMIN   2F08F   U	2F008	getypos		2F0D5	NEWINDEP	
TURNMENUOFF   273	2F019	UNIT>\$	82	2F0DB	MAKEPICT#	91
SaveLastEdit   S14	2F031	TURNMENUON	273	2F0E6	KERMOPEN	
StoIOPAR	2F034	TURNMENUOFF	273	2F0E7	InitIOEnv	
2F063	2F05E	SaveLastEdit	314	2F0E8	INDEPVAR	317
2F064   Sys@   166   2F0EF	2F062	StoIOPAR	181	2F0EC	ICMPDRPRTDRP	71
2F066   STOAPPLDATA   2F0FE   GETXMAX   316	2F063	StoPRTPAR		2F0EE	HXS#HXS	58
2F073       SWAPCOMPSWAP       86       2F0FF       GETXMIN       316         2F075       InitSysUI       2F100       GETYMIN       317         2F076       puretemp?       84       2F105       GDISPCENTER       318         2F07A       UM>U       82       2F106       GETINDEP       317         2F07B       U>nbr       82       2F107       GETPMIN&MAX       317         2F07C       UM#?       83       2F108       GETRHS         2F07D       UM%CH       83       2F109       GETXPOS         2F07E       UM%CH       83       2F10A       GetRes         2F07E       UM%T       83       2F10B       GETYMAX       317         2F08D       UM*       83       2F10E       GETYMAX       317         2F081       UM       83       2F10E       GETYMAX       317         2F081       UM       83       2F110       FINDVARS       86         2F082       UM       83       2F112       Echo\$NoChr00       302         2F085       UM<	2F064	Sys@	166	2F0EF	HXS<=HXS	59
2F075	2F066	STOAPPLDATA		2F0FE	GETXMAX	316
2F076       puretemp?       84       2F105       GDISPCENTER       318         2F07A       UM>U       82       2F106       GETINDEP       317         2F07B       U>nbr       82       2F107       GETPMIN&MAX       317         2F07C       UM#?       83       2F108       GETRHS         2F07D       UM%       83       2F109       GETXPOS         2F07E       UM%CH       83       2F10A       GetRes         2F07F       UM%T       83       2F10D       GETRES       317         2F080       UM*       83       2F10E       GETYMAX       317         2F081       UM+       83       2F10E       GETYMAX       317         2F081       UM+       83       2F110       FINDVARS       86         2F082       UM-       83       2F113       FNDALARM{}       86         2F083       UM/       83       2F11C       Echo\$NoChr00       302         2F085       UM<	2F073	SWAPcompSWAP	86	2F0FF	GETXMIN	316
2F07A         UM>U         82         2F106         GETINDEP         317           2F07B         U>nbr         82         2F107         GETPMIN&MAX         317           2F07C         UM#?         83         2F108         GETRHS           2F07D         UM%         83         2F109         GETXPOS           2F07E         UM%CH         83         2F10A         GetRes           2F07F         UM%T         83         2F10D         GETRES         317           2F080         UM*         83         2F10E         GETYMAX         317           2F081         UM+         83         2F10E         GETYMAX         317           2F081         UM+         83         2F10E         GETYMAX         317           2F082         UM-         83         2F110         FINDVARS         86           2F082         UM-         83         2F112         Echo\$NoChr00         302           2F083         UM         83         2F112         Echo\$NoChr00         302           2F086         UM         83         2F13D         CK+-         23           2F089         UM>?         83         2F153         CLKADJ* </td <td>2F075</td> <td>InitSysUI</td> <td></td> <td>2F100</td> <td>GETYMIN</td> <td>317</td>	2F075	InitSysUI		2F100	GETYMIN	317
2F07B       U>nbr       82       2F107       GETPMIN&MAX       317         2F07C       UM#?       83       2F108       GETRHS         2F07D       UM%       83       2F109       GETXPOS         2F07E       UM%CH       83       2F10A       GetRes         2F07F       UM%T       83       2F10D       GETRES       317         2F080       UM*       83       2F10E       GETYMAX       317         2F081       UM+       83       2F110       FINDVARS       86         2F082       UM-       83       2F113       FNDALARM{}       86         2F082       UM-       83       2F11C       Echo\$NoChr00       302         2F083       UM/       83       2F13C       DoOldMatrix       2         2F086       UM </td 83       2F13D       CK#-       23         2F087       UM=?       83       2F13F       DRAWLINE#3       92         2F088       UM>=?       83       2F153       CLKADJ*         2F08B       UMCHS       83       2F155       input{}       212         2F08C       UMCONV       82       2F158       THISCHAR       300 </td <td>2F076</td> <td>puretemp?</td> <td>84</td> <td>2F105</td> <td>GDISPCENTER</td> <td>318</td>	2F076	puretemp?	84	2F105	GDISPCENTER	318
2F07C       UM#?       83       2F108       GETRHS         2F07D       UM%       83       2F109       GETXPOS         2F07E       UM%CH       83       2F10A       GetRes         2F07F       UM%T       83       2F10D       GETRES       317         2F080       UM*       83       2F10E       GETYMAX       317         2F081       UM+       83       2F110       FINDVARS       86         2F082       UM-       83       2F113       FNDALARM{}         2F083       UM/       83       2F11C       Echo\$NoChr00       302         2F085       UM       83       2F13C       DooldMatrix         2F086       UM       83       2F13D       CK#-       23         2F087       UM=?       83       2F13F       DRAWLINE#3       92         2F088       UM>=?       83       2F142       DONewMatrix         2F089       UM>?       83       2F153       CLKADJ*         2F08B       UMCHS       83       2F155       input{}       212         2F08C       UMCONV       82       2F158       THISCHAR       300         2F08E       UMMI	2F07A	UM>U	82	2F106	GETINDEP	317
2F07D       UM%       83       2F109       GETXPOS         2F07E       UM%CH       83       2F10A       GetRes         2F07F       UM%T       83       2F10D       GETRES       317         2F080       UM*       83       2F10E       GETYMAX       317         2F081       UM+       83       2F110       FINDVARS       86         2F082       UM-       83       2F113       FNDALARM{}         2F083       UM/       83       2F11C       Echo\$NOChr00       302         2F085       UM       83       2F11C       Echo\$NOChr00       302         2F085       UM       83       2F13C       DOOldMatrix         2F086       UM       83       2F13D       CK#-       23         2F087       UM=?       83       2F13F       DRAWLINE#3       92         2F088       UM>=?       83       2F142       DoNewMatrix         2F089       UM>?       83       2F153       CLKADJ*         2F08B       UMCHS       83       2F155       input{}       212         2F08C       UMCONV       82       2F158       THISCHAR       300         2F	2F07B	U>nbr	82	2F107	GETPMIN&MAX	317
2F07E         UM%CH         83         2F10A         GetRes           2F07F         UM%T         83         2F10D         GETRES         317           2F080         UM*         83         2F10E         GETYMAX         317           2F081         UM+         83         2F110         FINDVARS         86           2F082         UM-         83         2F113         FNDALARM{}           2F083         UM/         83         2F11C         Echo\$NoChr00         302           2F085         UM         83         2F13C         DoOldMatrix         250           2F086         UM         83         2F13D         CK#-         23           2F087         UM=?         83         2F13F         DRAWLINE#3         92           2F088         UM>=?         83         2F142         DoNewMatrix         2F089           2F089         UM>?         83         2F153         CLKADJ*         2F12           2F08B         UMCHS         83         2F155         input{}         212           2F08C         UMCONV         82         2F158         THISCHAR         300           2F08E         UMMAX         83         <	2F07C	UM#?	83	2F108	GETRHS	
2F07F       UM%T       83       2F10D       GETRES       317         2F080       UM*       83       2F10E       GETYMAX       317         2F081       UM+       83       2F110       FINDVARS       86         2F082       UM-       83       2F113       FNDALARM{}         2F083       UM/       83       2F11C       Echo\$NoChr00       302         2F085       UM       83       2F13C       DoOldMatrix         2F086       UM       83       2F13D       CK#-       23         2F087       UM=?       83       2F13F       DRAWLINE#3       92         2F088       UM>=?       83       2F142       DONewMatrix         2F089       UM>?       83       2F153       CLKADJ*         2F08A       UMABS       83       2F154       input{}       212         2F08C       UMCONV       82       2F158       THISCHAR       300         2F08D       UMCOS       83       2F15A       CHOOSE_INIT         2F08E       UMMAX       83       2F15B       CLEARMENU       293         2F08F       UMMIN       83       2F15E       Clr16       277 <td>2F07D</td> <td>UM%</td> <td>83</td> <td>2F109</td> <td>GETXPOS</td> <td></td>	2F07D	UM%	83	2F109	GETXPOS	
2F080       UM*       83       2F10E       GETYMAX       317         2F081       UM+       83       2F110       FINDVARS       86         2F082       UM-       83       2F113       FNDALARM{}         2F083       UM/       83       2F11C       Echo\$NoChr00       302         2F085       UM       83       2F13C       DoOldMatrix         2F086       UM       83       2F13D       CK#-       23         2F087       UM=?       83       2F13F       DRAWLINE#3       92         2F088       UM>=?       83       2F142       DONewMatrix         2F089       UM>?       83       2F153       CLKADJ*         2F08A       UMABS       83       2F154       input\$       212         2F08B       UMCON       82       2F155       input{}       212         2F08C       UMCONV       82       2F158       THISCHAR       300         2F08E       UMMAX       83       2F15B       CLEARMENU       293         2F08F       UMMIN       83       2F15E       Clr16       277	2F07E	UM%CH	83	2F10A	GetRes	
2F081       UM+       83       2F110       FINDVARS       86         2F082       UM-       83       2F113       FNDALARM{}       81         2F083       UM/       83       2F11C       Echo\$NoChr00       302         2F085       UM       83       2F13C       DoOldMatrix       28         2F086       UM       83       2F13D       CK#-       23         2F087       UM=?       83       2F13F       DRAWLINE#3       92         2F088       UM>=?       83       2F142       DoNewMatrix         2F089       UM>?       83       2F153       CLKADJ*         2F08A       UMABS       83       2F154       input\$       212         2F08B       UMCHS       83       2F155       input{}       212         2F08C       UMCONV       82       2F158       THISCHAR       300         2F08D       UMCOS       83       2F15A       CHOOSE_INIT         2F08E       UMMAX       83       2F15B       CLEARMENU       293         2F08F       UMMIN       83       2F15E       Clr16       277	2F07F	UM%T	83	2F10D	GETRES	317
2F082       UM-       83       2F113       FNDALARM{}         2F083       UM/       83       2F11C       Echo\$NoChr00       302         2F085       UM<=?	2F080	UM*	83	2F10E	GETYMAX	317
2F083         UM/         83         2F11C         Echo\$noChr00         302           2F085         UM<=?	2F081	UM+	83	2F110	FINDVARS	86
2F085       UM<=?	2F082	UM-	83	2F113	<pre>FNDALARM{ }</pre>	
2F086     UM </td 83     2F13D     CK#-     23       2F087     UM=?     83     2F13F     DRAWLINE#3     92       2F088     UM>=?     83     2F142     DONewMatrix       2F089     UM>?     83     2F153     CLKADJ*       2F08A     UMABS     83     2F154     input\$     212       2F08B     UMCHS     83     2F155     input{}     212       2F08C     UMCONV     82     2F158     THISCHAR     300       2F08D     UMCOS     83     2F15A     CHOOSE_INIT       2F08E     UMMAX     83     2F15B     CLEARMENU     293       2F08F     UMMIN     83     2F15E     Clr16     277	2F083	UM/	83	2F11C	Echo\$NoChr00	302
2F087     UM=?     83     2F13F     DRAWLINE#3     92       2F088     UM>=?     83     2F142     DONewMatrix       2F089     UM>?     83     2F153     CLKADJ*       2F08A     UMABS     83     2F154     input\$     212       2F08B     UMCHS     83     2F155     input{}     212       2F08C     UMCONV     82     2F158     THISCHAR     300       2F08D     UMCOS     83     2F15A     CHOOSE_INIT       2F08E     UMMAX     83     2F15B     CLEARMENU     293       2F08F     UMMIN     83     2F15E     Clr16     277	2F085	UM <= ?	83	2F13C	DoOldMatrix	
2F088       UM>=?       83       2F142       DoNewMatrix         2F089       UM>?       83       2F153       CLKADJ*         2F08A       UMABS       83       2F154       input\$       212         2F08B       UMCHS       83       2F155       input{}       212         2F08C       UMCONV       82       2F158       THISCHAR       300         2F08D       UMCOS       83       2F15A       CHOOSE_INIT         2F08E       UMMAX       83       2F15B       CLEARMENU       293         2F08F       UMMIN       83       2F15E       Clr16       277	2F086	UM </td <td>83</td> <td>2F13D</td> <td>CK#-</td> <td>23</td>	83	2F13D	CK#-	23
2F089     UM>?     83     2F153     CLKADJ*       2F08A     UMABS     83     2F154     input\$     212       2F08B     UMCHS     83     2F155     input{}     212       2F08C     UMCONV     82     2F158     THISCHAR     300       2F08D     UMCOS     83     2F15A     CHOOSE_INIT       2F08E     UMMAX     83     2F15B     CLEARMENU     293       2F08F     UMMIN     83     2F15E     Clr16     277	2F087	UM=?	83	2F13F	DRAWLINE#3	92
2F08A     UMABS     83     2F154     input\$     212       2F08B     UMCHS     83     2F155     input{}     212       2F08C     UMCONV     82     2F158     THISCHAR     300       2F08D     UMCOS     83     2F15A     CHOOSE_INIT       2F08E     UMMAX     83     2F15B     CLEARMENU     293       2F08F     UMMIN     83     2F15E     Clr16     277	2F088	UM>=?	83	2F142	DoNewMatrix	
2F08B     UMCHS     83     2F155     input{}     212       2F08C     UMCONV     82     2F158     THISCHAR     300       2F08D     UMCOS     83     2F15A     CHOOSE_INIT       2F08E     UMMAX     83     2F15B     CLEARMENU     293       2F08F     UMMIN     83     2F15E     Clr16     277	2F089	UM>?	83	2F153		
2F08C         UMCONV         82         2F158         THISCHAR         300           2F08D         UMCOS         83         2F15A         CHOOSE_INIT           2F08E         UMMAX         83         2F15B         CLEARMENU         293           2F08F         UMMIN         83         2F15E         Clr16         277	2F08A	UMABS	83			212
2F08D         UMCOS         83         2F15A         CHOOSE_INIT           2F08E         UMMAX         83         2F15B         CLEARMENU         293           2F08F         UMMIN         83         2F15E         Clr16         277	2F08B	UMCHS	83			212
2F08E         UMMAX         83         2F15B         CLEARMENU         293           2F08F         UMMIN         83         2F15E         Clr16         277	2F08C	UMCONV	82	_		300
2F08F UMMIN 83 2F15E Clr16 277	2F08D	UMCOS	83			
2F08F UMMIN 83 2F15E Clr16 277	2F08E	UMMAX	83			293
2F090 UMSI 82 2F162 CHECKPICT 316	2F08F	UMMIN				
	2F090	UMSI	82	2F162	CHECKPICT	316

Addr.	Name	Page	Addr.	Name	Page
2F163	CHECKPVARS	316	2F223	%>TAG	61
2F16D	Blank\$	48	2F23E	DOSTOSYSF	175
2F177	AllowPrlcdCl		2F244	COERCE&CKSGN	22
2F178	ALARMS@	174	2F24E	LISTRCL	166
2F179	AdjEdModes		2F257	OCRC%	179
2F17A	APPprompt2		2F258	PICTRCL	316
2F17E	2HXSLIST?	22	2F259	RCLSYSF	175
2F180	1REV	30	2F25A	RCLSYSF2	175
2F190	DcompWidth@	51	2F25B	RCLUSERF	176
2F191	!DcompWidth	51	2F25C	RCLUSERF2	176
2F192	DoNewEqw		2F25D	SETROMPART	
2F193	UobROT	77	2F25F	STOSYSF	175
2F194	CMD_PLUS2	301	2F260	STOSYSF2	176
2F195	CMD_PLUS3	302	2F261	STOUSERF	176
2F196	RCL_CMD	300	2F262	STOUSERF2	176
2F197	RCL_CMD2	300	2F265	UPDIR	169
2F198	STO_CMD_MODE	301	2F266	USER\$>TAG	61
2F199	RCL_CMD_MODE	301	2F267	VARSIZE	179
2F19A	ViewLevel1	310	2F268	Wait/GetKey	207
2F19B	OngoingText?		2F292	XEQIOBACKUP	
2F19E	DispCommandLine	312,274	2F296	XEQORDER	168
2F19F	?DispCommandLine	312,274	2F297	XEQPURGEPICT	318
2F1A1	ErrorHandled?	·	2F2A3	XEQRCL	166
2F1A3	SysErrorTrapAction		2F2A7	XEQSETLIB	100
2F1A5	AskQuestion	282	2F2A9	XEQSHOWLS	88
2F1A7	CHARSEDIT	50	2F2D4	dowait	172
2F1A8	EditFont		2F2DA	AlgCharEdit	
2F1A9	EDITF		2F2DB	DOTEXTINFO	312
2F1AB	Date>hxs13	173	2F2DC	ClearSelection	306
2F1AC	StrEdit		2F2DD	DoFarBS	303
2F1AD	CharEdit		2F2DE	DoFarDel	303
2F1AE	ObEdit	311	2F2DF	EditSelect	309
2F1AF	AlgObEdit	311	2F2E0	ViewEditGrob	313
2F1BF	Decomp%Short	54	2F2E1	SELECT.LINE	307
2F1C6	DROP3PICK	109	2F2E2	SELECT.LINEEND	307
2F205	laMGET0		2F2E3	EVAL.LINE	309
2F215	CircleB	93	2F2E4	DO>BEG	305
2F216	CircleG1	93	2F2E5	DO>END	306
2F217	CircleG2	93	2F2E6	GOTOLABEL	306
2F218	CircleW	93	2F2E7	SELECT.FONT	313
2F219	CircleXor	93	2F2E8	DOFIND	308
2F21A	Dither	95	2F2E9	DOREPL	308
2F21B	ToGray	95	2F2EA	DONEXT	308
2F21C	Lift		2F2EB	DOREPLACE	308
2F21D	ViewObject	280	2F2EC	DOREPLACE/NEXT	308
2F21E	ViewStrObject	280	2F2ED	REPLACEALL	309
2F21F	ViewGrobObject	280	2F2EE	DO <skip< td=""><td>305</td></skip<>	305
2F222	#1+ROT	24	2F2EF	DO>Skip	305

2F163 - 2F365 573

Addr.	Name	Page	Addr. Name		Page
2F2F0	DO <del< td=""><td>303</td><td>2F336</td><td>FindNext</td><td></td></del<>	303	2F336	FindNext	
2F2F1	DO>Del	303	2F337	FixRRP	
2F2F2	FindStrInCmd	308	2F338	GetChkPRTPAR	
2F2F3	GET.W->	308	2F339	GetEqN	318
2F2F4	GET.W<-	308	2F33A	GetKermPkt#	
2F2F5	PUT_FONTE	313	2F33B	GETKP	
2F2F6	GET_CUR_FONT.EXT	312	2F33C	getmatchtok	51
2F2F7	PUT_STYLE	313	2F33D	GETPARAM	316
2F2F8	EXEC_CMD	309	2F33E	GETSCALE	317
2F2F9	DODEL.L	303	2F33F	GETSERIAL	
2F2FA	CMD_COPY.SBR	307	2F340	GETYPOS	
2F2FB	EVAL.SELECTION	309	2F341	GraphicExit	
2F2FC	REPLACEALLNOSCREEN	309	2F342	GROB+#	91
2F2FF	OpenIO		2F343	IncrLAMPKNO	
2F300	DispILPrompt	274	2F344	InputLAttn	
2F312	OpenUartClr		2F345	InputLEnter	
2F313	OpenUart?Clr		2F346	IOCheckReal	
2F314	RCLALARM%	174	2F347	JUMPBOT	280
2F315	!#1+IF <dim-1< td=""><td></td><td>2F348</td><td>JUMPLEFT</td><td>280</td></dim-1<>		2F348	JUMPLEFT	280
2F316	!#1-IF>0		2F349	JUMPRIGHT	280
2F318	1GETLAMSWP1+	119	2F34A	JUMPTOP	280
2F319	ACK_INIT		2F34B	KDispRow2	
2F31A	APNDCRLF	47, 181	2F34C	KDispStatus2	
2F31B	BlankDA2	277	2F34D	KINVISLF	181
2F31C	BlankDA2a	277	2F34E	KVIS	181
2F31D	BOTROW	279	2F34F	KVISLF	181
2F31E	BUILDKPACKET		2F350	'LamKPSto	
2F31F	C%>#	22	2F351	LASTPT?	
2F320	CHECKHEIGHT	90	2F352	LEFTCOL	279
2F321	CkChr00	55	2F353	LINECHANGE	
2F324	CKGROBFITS	90	2F354	List	
2F325	ClrServMode		2F355	MAKEPVARS	316
2F326	CMDSTO	314	2F356	metatail	78
2F327	convertbase		2F357	newBASE	
2F328	CROSSMARKON		2F358	NEWMARK	
2F329	Date>d\$	173	2F359	NEXTRRPOB	
2F32A	DECODE		2F35A	NEXTSTEP	
2F32B	DISPCOORD2	282	2F35B	NUMSOLVE	
2F32C	DRAWBOX#	92	2F35C	OB>BAKcode	
2F32D	drax		2F35D	OpenIOPrt	
2F32E	DROPDEADTRUE	130	2F35E	PLOTERR	
2F32F	DropSysErr\$		2F35F	PlotOneMore?	
2F330	ENCODE		2F360	PREMARKON	
2F331	ENCODE1PKT		2F361	PrintGrob	
2F332	EQCURSOR?		2F362	PRINTXNLF	
2F333	EXCHINITPK		2F363	PtoR	
2F334	Extobcode		2F364	PUTSERIAL	
2F335	FcnUtilEnd		2F365	PUTXMAX	316

Addr.	Name	Page	Addr.	Name	Page
2F366	PUTXMIN	316	2F94C	%1	28
2F367	PUTYMAX	317	2F961	%2	28
2F368	PUTYMIN	317	2F976	%3	28
2F369	RECORDX&YC%		2F98B	%4	28
2F36A	REMAP		2F9A0	%5	28
2F36B	RIGHTCOL	279	2F9B5	86	28
2F36C	Rows8-15	279	2F9CA	%7	28
2F36D	SCROLLDOWN	279	2F9DF	%8	28
2F36E	SCROLLLEFT	280	2F9F4	89	28
2F36F	SCROLLRIGHT	280	2FA09	%-1	28
2F370	SCROLLUP	279	2FA1E	%-2	28
2F371	SENDACK		2FA33	%-3	28
2F372	SENDEOT		2FA48	%-4	28
2F373	SENDERROR		2FA5D	%-5	28
2F374	SENDNAK		2FA72	%-6	28
2F375	SENDNULLACK		2FA87	%-7	28
2F376	SENDPKT		2FA9C	%-8	28
2F377	SendSetup		2FAB1	%-9	28
2F378	SetCursor	304	2FAC6	%PI	28
2F379	SetDA123NoCh	276	2FADB	%%PI	30
2F37A	SetDA20KTemp	275	2FAF5	%MAXREAL	30
2F37B	SetIOPARErr	158, 181	2FB0A	%-MAXREAL	28
2F37C	SETLOOPENV	100, 101	2FB1F	%MINREAL	28
2F37D	SetServMode		2FB34	%-MINREAL	28
2F37E	SORTASLOW	172	2FB49	%%O	30
2F37F	STOALM	174	2FB63	%%1	30
2F380	SysST0	167	2FB7D	%%2	30
2F381	TOD>t\$	173	2FB97	%%3	30
2F382	TOGGLELINE#3	92	2FBB1	%%4	30
2F383	TOP16	279	2FBCB	%%5	30
2F384	TOP8	279	2FBE5	%%.1	30
2F385	TOPROW	279	2FBFF	%%.5	30
2F386	TRPACKETFAIL	210	2FC19	%%10	30
2F387	UARTBUFLEN		2FC7D	%1200	29
2F388	VerifyTOD	172	2FC7D 2FC92	%2400	29
2F389	VERSTRING	181	2FC3Z 2FCA7	%4800	29
2F38A	WINDOWBOT?	280	2FCBC	%9600 %9600	29
2F38B	WINDOWLEFT?	280	2FCDC 2FCD1	%15360	29
2F38C	WINDOWLEFT: WINDOWRIGHT?	280	2FCE1	%11	28
2F38D	WINDOWRIGHT: WINDOWTOP?	280	2FCE0 2FCFB	%12	28
2F38E		77	2FCFB 2FD10	%13	29
	xnsgeneral	77	2FD10 2FD25	%13 %14	29 29
2F38F 2F390	xsngeneral	11	2FD25 2FD3A	%15	29 29
	xssgeneral	176		%15 %16	29 29
2F3A9	STOALLE	176	2FD4F 2FD64	%17	29 29
2F3AA	STOALLF2				
2F3B3	AsnKey	208	2FD79	%18 %10	29
2F458	SETIVLERR	158	2FD8E	%19 %20	29
2F937	%0	28	2FDA3	%20	29

2F366 - 30663 575

2PDBB   \$21	Addr.	Name	Page	Addr.	Name	Page
2FDE2	2FDB8	<b>%21</b>	29	302A1	%<=	35
2FDFF	2FDCD	%22	29	302AC	%=	35
2FEOC   \$25   29   302DB   \$\$ABS   34     2FE21   \$26   29   302EB   \$\$ABS   32     2FE36   \$27   29   302FB   \$\$CHS   34     2FE4B   \$28   29   303DB   \$\$CHS   32     2FE60   \$29   29   3031B   \$\$MANTISSA   32     2FE60   \$29   29   3031B   \$\$MANTISSA   32     2FE75   \$30   29   3032B   \$\$\$**   33     2FE9A   \$31   29   3033A   \$\$\$*-   33     2FE9F   \$32   29   30346   \$\$\$**-   31     2FEB4   \$33   29   3035C   \$\$\$*-   31     2FED5   \$35   29   30385   \$\$\$**   33     2FF9B   \$35   29   30385   \$\$\$**   33     2FF9B   \$35   29   30385   \$\$\$**   31     2FFAC   \$\$\$**   \$31   303A   \$	2FDE2	%23	29	302B7	%<>	35
2FE21	2FDF7	%24	29	302C2	%SGN	32
2FE36   %27   29   302FB   %%CHS   32   2FE6B   %29   29   3031B   %MANTISSA   32   2FE75   %30   29   3032E   %%+   33   32   333A   %%-   33   34   34   34   34   34   34   3	2FE0C	%25	29	302DB	%%ABS	34
2FE4B	2FE21	%26	29	302EB	%ABS	32
2FE60   %29   29   3031B	2FE36	%27	29	302FB	%%CHS	34
2FE75	2FE4B	%28	29	3030B	%CHS	32
2FE8A       \$31       29       3033A       *%-       33         2FEBF       \$32       29       30346       %>%-       31         2FEB4       *33       29       3035F       %+       31         2FEDE       *35       29       3036C       %-       31         2FEDE       *35       29       30385       %**       33         2FFDB       ***       31       303A7       **       31         2FFDB       ***       31       303B4       **OF       33         2FFDB       ***       31       303B4       **OF       33         30017       ***ETDEG       178       303B9       */       32         2FFEB       **SETRAD       178       303F6       %*T       33         30017       **PI/180       30       3044B       %*CH       33         30017       *PI/180       30       3044B       %*CH       33         30017       **PI/180       30       3044B       %*T       34         30040       *R>D       33       3045B       %*1/       34         30077       **HMS       172       3046C       *NROOT	2FE60	%29	29	3031B	%MANTISSA	32
2FE9F	2FE75	%30	29	3032E	%%+	33
2FEB4	2FE8A	%31	29	3033A	% % <b>–</b>	33
2FEC9       %34       29       3036C       %-       31         2FEDE       %35       29       30385       %**       33         2FF9B       %*>%       31       30387       **       31         2FFAC       %>%       31       303B4       %OF       33         2FFDB       SETDEG       178       303D3       %*/       34         2FFDB       SETRAD       178       303E9       %/       32         2FFFF       SETGRAD       178       303E9       %/       32         30017       PI/180       30       3044A       %%^       34         30040       %R>D       33       3045B       %^       32         3005E       %+MMS       172       3046C       %NROOT       33         3008B       %HMS+       172       3047D       %*1/       34         3008B       %HMS+       172       3049A       %1/       32         300E7       %MAX       34       304D5       %*SQRT       34         30DE0       %MAX       33       304F4       %SQRT       32         30123       %0       35       305D7       %EXP	2FE9F	%32	29	30346	%>%%-	31
2FEDE       %35       29       30385       %**       31         2FF9B       %5%       31       303A7       %*       31         2FFAC       %>%*       31       303B4       %OF       33         2FFBD       SETDEG       178       303B4       %OF       34         2FFDB       SETRAD       178       303E9       %/       32         2FFFF       SETGRAD       178       303E9       %/       32         300D       %DR       33       3041B       %CH       33         30017       PI/180       30       3044A       %*       34         30040       %R>D       33       3045B       %^       32         3005E       %HMS       172       3046C       %ROOT       33         30077       %HMS>       172       3047D       %*1/       34         3008B       %HMS+       172       3049A       %1/       32         300E0       %MAX       34       304D5       %\$QRT       34         3012       %%0       35       3051A       %EXP       32         30123       %0       35       3051A       %EXP       3	2FEB4	%33	29	3035F	%+	31
2FF9B         %% % %         31         303A7         %*         31           2FFAC         % % %         31         303B4         %OF         33           2FFDD         SETDEG         178         303D3         %*/         34           2FFDB         SETRAD         178         303E9         %/         32           2FFEF         SETGRAD         178         303F6         %T         33           3000D         %D>R         33         3041B         %CH         33           30017         PI/180         30         3044A         %*^         34           30040         %R>D         33         3045B         %^         32           30040         %R>D         33         3045B         %^         32           3005E         %HMS         172         3047D         %*1/         34           3008B         %HMS+         172         3049A         %1/         32           3008B         %HMS+         172         3049A         %1/         32           300F9         %MIN         33         304E1         %%SQRT         32           3012         %%0         35         30507	2FEC9	%34	29	3036C	% –	31
2FFAC         %>%%         31         303B4         %OF         33           2FFBD         SETDEG         178         303D3         %/         34           2FFDB         SETRAD         178         303E9         %/         32           2FFEF         SETGRAD         178         303E9         %/         32           300D         %DR         33         3041B         %CH         33           30017         PI/180         30         3044A         %*^         34           30040         %R>D         33         3045B         %*         32           30040         %R>D         33         3045B         %*         32           30055         %HMS         172         3046C         %NROOT         33           30077         %HMS>         172         3044D         %*\$1/         32           30088         %HMS+         172         30489         %>%\$1/         32           300C7         %MAX         34         304D5         %*\$SQRT         34           3012         %MAX         33         304E1         %>%*\$SQRT         32           30112         %80         35         3051A <td>2FEDE</td> <td>%35</td> <td>29</td> <td>30385</td> <td>%%<b>*</b></td> <td>33</td>	2FEDE	%35	29	30385	%% <b>*</b>	33
2FFBD         SETDEG         178         303D3         %%/         34           2FFDB         SETRAD         178         303E9         %/         32           2FFFF         SETGRAD         178         303F6         %T         33           30010         %D>R         33         3041B         %CH         33           30017         PI/180         30         3044A         %%^         34           30040         %R>D         33         3045B         %^         32           3005E         %>HMS         172         3046C         %NROOT         33           30077         *HMS>         172         3047D         %*1/         32           30083         *HMS-         172         3049A         %1/         32           300C7         *\$MAX         34         304D5         %*\$SQRT         34           300E0         *MAX         33         304F4         %SQRT         32           30112         %0         35         3051A         *EXP         32           30145         %*0=         35         3051A         *EXP         32           30145         %0=         35         30546 <td>2FF9B</td> <td>%%&gt;%</td> <td>31</td> <td>303A7</td> <td>%*</td> <td>31</td>	2FF9B	%%>%	31	303A7	%*	31
2FFDB         SETRAD         178         303E9         %/         32           2FFEF         SETGRAD         178         303F6         %T         33           3000D         %D>R         33         3041B         %CH         33           30017         PI/180         30         3044A         %*^         34           30040         %R>D         33         3045B         %^         32           3005E         %HMS         172         3046C         %NROOT         33           30077         %HMS>         172         3047D         %*1/         34           3008B         %HMS+         172         30489         %>%*1/         32           300C7         %MAX         34         304D5         %*SQRT         34           300E0         %MAX         33         304E1         %>%*SQRT         32           30112         %*0<	2FFAC	%>%%	31	303B4	%OF	33
2FFDB         SETRAD         178         303E9         \$/         32           2FFEF         SETGRAD         178         303F6         %T         33           3000D         &D>R         33         3041B         %CH         33           30017         PI/180         30         3044A         %*^         34           30040         %R>D         33         3045B         %^         32           3005E         %>HMS         172         3046C         %NROOT         33           30077         %HMS>         172         3047D         %*1/         34           3008B         %HMS+         172         3049A         %1/         32           30077         %MAX         34         304D5         %*\$SQRT         32           30073         %MAX         34         304D5         %*\$SQRT         34           30060         %MAX         33         304F4         %SQRT         32           30112         %*0<	2FFBD	SETDEG	178	303D3	%%/	34
3000D %D>R 33 3041B %CH 33 30017 PI/180 30 3044A %%^ 34 30040 %R>D 33 3045B %^ 32 3005E %>HMS 172 3046C %NROOT 33 30077 %HMS> 172 3047D %%1/ 34 3008B %HMS+ 172 30489 %>%%1/ 32 30083 %HMS- 172 3049A %1/ 32 300C7 %%MAX 34 304D5 %%SQRT 34 300E0 %MAX 33 304E1 %>%%SQRT 32 301E2 %0>= 35 3051A %EXP 32 30145 %%0- 35 30559 %LN 32 30156 %0- 35 3056C %LOG 32 30184 %0> 35,135 30592 %LNP1 34 3018A %0<> 35,135 30507 %MOD 32 301E2 %0>= 35 305DA %SIN 32 3020A %%< 35 305DA %SIN 32 3020A %%< 35 305DA %SIN 32 3020A %%>= 35 306C2 %%SINDEG 34 302EB %>= 35 30642 %%COS 34 30280 %%>= 35 30642 %%COS 34 30280 %%>= 35 30642 %%COS 34	2FFDB	SETRAD	178		% /	32
30017       PI/180       30       3044A       %%^       34         30040       %R>D       33       3045B       %^       32         3005E       %>HMS       172       3046C       %NROOT       33         30077       %HMS>       172       30489       %>%1/       34         3008B       %HMS+       172       3049A       %1/       32         300C7       %MAX       34       304D5       %*SQRT       34         300E0       %MAX       33       304E1       %>%*SQRT       32         3012       %%0       35       30507       %*EXP       32         3012       %%0       35       3051A       *EXP       32         30145       %%0=       35       3051A       *EXP       32         30145       %%0=       35       3052D       *EXPM1       32         30146       %0       35       30546       %*LN       34         30173       %%0       35       30559       %LN       32         30184       %0       35       30559       %LN       32         30186       %%0<	2FFEF	SETGRAD	178	303F6	%T	33
30040 %R>D 33 3045B %^ 32 3005E %>HMS 172 3046C %NROOT 33 30077 %HMS> 172 3047D %%1/ 34 3008B %HMS+ 172 30489 %>%1/ 32 30067 %*MAX 34 304D5 %*SQRT 34 30060 %MAX 33 304E1 %>%SQRT 32 30069 %MIN 33 304E1 %>%SQRT 32 30112 %%0< 35 30507 %*EXP 34 30123 %0< 35 3051A %EXP 32 30145 %%0= 35 3052D %EXPM1 32 30156 %0= 35 30546 %%IN 34 30173 %%0> 35 3056C %LOG 32 30184 %0> 35 3056C %LOG 32 30184 %0> 35 3056C %LOG 32 30184 %0> 35 3057F %%INP1 34 3018A %0<> 35,135 30592 %LNP1 32 3016C %%0<> 35,135 30592 %LNP1 32 3016C %%0<= 35 30557 %MOD 32 3016C %%0<= 35 30557 %MOD 32 3016C %%0<= 35 30557 %MOD 32 3016C %%0<= 35 30551 %SIN 32 3020A %%< 35 30551 %SIN 32 3020A %%< 35 3056C %SIN 32 3020A %%< 35 3056C %SIN 32 3020A %%< 35 3056C %SIN 32 3020A %%< 35 3056D %SIN 32 3020A %%< 35 306DA %SIN 34 3025C %< 35 306DB %SIN 34 3025C %< 35 306DB %SIN 34 3025C %< 35 306DB %SIN 34 3026A %%> 35 306DB %COS 32 30280 %%>= 35 306BB %COS 32 30280 %%>= 35 306BB %COS 32 30280 %%>= 35 306BB %COS 34 3028B %>= 35 306BB %COSDEG 34	3000D	%D>R	33	3041B	%CH	33
3005E   %>HMS   172   3046C   %NROOT   33   30077   %HMS>   172   3047D   %%1/   34   34   3008B   %HMS+   172   30489   %>%%1/   32   300B3   %HMS-   172   3049A   %1/   32   300C7   %MAX   34   304D5   %%SQRT   34   300E0   %MAX   33   304E1   %>%SQRT   32   30112   %%0<   35   30507   %%EXP   34   30112   %%0<   35   30507   %%EXP   34   30123   %0<   35   3051A   %EXP   32   30145   %%0=   35   3052D   %EXPM1   32   30145   %%0=   35   3052D   %EXPM1   32   30145   %%0=   35   30546   %%LN   34   30173   %%0>   35   30546   %%LN   34   30173   %%0>   35   30559   %LN   32   30184   %0>   35   3056C   %LOG   32   30184   %0<   35   3057F   %%LNP1   34   301BA   %0<>   35   3057F   %%LNP1   34   301BA   %0<>   35   30552   %LNP1   32   301E2   %0>=   35   30553   %ALOG   32   301E2   %0>=   35   30554   %SIN   32   301E2   %0>=   35   305DA   %SIN   32   3026A   %%<   35   305DA   %SIN   32   3026A   %%>   35   30602   %\$SINBEG   34   3025C   %<   35   30602   %\$SINBEG   34   3028B   %>=   35   30642   %\$COS   34   3028B   %>=   35   30653   %\$COSDEG   34   3028B   %=   35   30653   %\$COSDEG   3	30017	PI/180	30	3044A	%%^	34
30077     %HMS>     172     3047D     %%1/     34       3008B     %HMS+     172     30489     %>%%1/     32       300B3     %HMS-     172     3049A     %1/     32       300C7     %MAX     34     304D5     %SQRT     34       300E0     %MAX     33     304E1     %>%SQRT     32       300F9     %MIN     33     304F4     %SQRT     32       30112     %%0     35     30507     %EXP     34       30123     %0     35     3051A     %EXP     32       30145     %%0=     35     3052D     %EXPM1     32       30156     %0=     35     30546     %LN     34       30173     %%0>     35     30559     %LN     32       30184     %0>     35     3056C     %LOG     32       301A6     %%0     35     3057F     %LNP1     34       301BA     %0<>     35     305A5     %ALOG     32       301E2     %0>=     35     305A5     %ALOG     32       301E2     %0>=     35     305A5     %ALOG     32       301E6     %%0<=	30040	%R>D	33	3045B	%^	32
3008B     %HMS+     172     30489     %>%%1/     32       300B3     %HMS-     172     3049A     %1/     32       300C7     %MAX     34     304D5     %*SQRT     34       300E0     %MAX     33     304E1     %>%*SQRT     32       300F9     %MIN     33     304F4     %SQRT     32       30112     %*0     35     30507     %*EXP     34       30123     %0     35     3051A     %EXP     32       30145     %*0=     35     3052D     %EXPM1     32       30156     %0=     35     30546     %LN     34       30173     %*0>     35     30559     %LN     32       30184     %0>     35     3056C     %LOG     32       30184     %0>     35     3057F     %LNP1     34       301BA     %0<>     35     3057F     %LNP1     32       301E6     %0<=	3005E	%>HMS	172	3046C	%NROOT	33
300B3       %HMS-       172       3049A       %1/       32         300C7       %%MAX       34       304D5       %%SQRT       34         300E0       %MAX       33       304E1       %>%%SQRT       32         300F9       %MIN       33       304F4       %SQRT       32         30112       %%0       35       30507       %%EXP       34         30123       %0       35       3051A       %EXP       32         30145       %0=       35       3052D       %EXPM1       32         30156       %0=       35       30546       %kln       34         30173       %%0>       35       3056C       %LOG       32         30184       %0>       35       3056C       %LOG       32         30184       %0>       35       3057F       %klnP1       34         301BA       %0<>       35       3057F       %klnP1       32         301EC       %0>=       35       305A5       %LNP1       32         301E2       %0>=       35       305A5       %ALOG       32         301E6       %0<=	30077	%HMS>	172	3047D	%%1/	34
300C7       %*MAX       34       304D5       %*SQRT       34         300E0       %MAX       33       304E1       %>%*SQRT       32         300F9       %MIN       33       304F4       %SQRT       32         30112       %*0       35       30507       %*EXP       34         30123       %0       35       3051A       %EXP       32         30145       %*0=       35       3052D       %EXPM1       32         30156       %0=       35       30546       %*LN       34         30173       %*0>       35       30559       %LN       32         30184       %0>       35       3056C       %LOG       32         301A6       %*0<>/td>       35       3057F       %*LNP1       34         301BA       %0<>/td>       35,135       30592       %LNP1       32         301CE       %*0>=       35       305A5       %ALOG       32         301E2       %0>=       35       305DA       %SIN       32         302A       %*<	3008B	%HMS+	172	30489	%>%%1/	32
300E0       %MAX       33       304E1       %>%\$SQRT       32         300F9       %MIN       33       304F4       %SQRT       32         30112       %%0       35       30507       %*EXP       34         30123       %0       35       3051A       *EXP       32         30145       %%0=       35       3052D       *EXPM1       32         30156       %0=       35       30546       %*LN       34         30173       %%0>       35       30559       %LN       32         30184       %0>       35       3056C       %LOG       32         301A6       %%0<>/td>       35       3057F       %*LNP1       34         301BA       %0<>/td>       35, 135       30592       %LNP1       32         301CE       %%0>=       35       305A5       %ALOG       32         301E2       %0>=       35       305DA       %SIN       32         302A       %       35       305DA       %SIN       32         302A       %       35       306D2       %SINDEG       34         302A       %       35       30612       %SINRAD	300B3	%HMS-	172	3049A	%1/	32
300F9       %MIN       33       304F4       %SQRT       32         30112       %%0       35       30507       %EXP       34         30123       %0       35       3051A       %EXP       32         30145       %%0=       35       3052D       %EXPM1       32         30156       %0=       35       30546       %kIN       34         30173       %%0>       35       30559       %LN       32         30184       %0>       35       3056C       %LOG       32         301A6       %%0<>       35       3057F       %*LNP1       34         301BA       %0<>       35,135       30592       %LNP1       32         301CE       %%0>=       35       305A5       %ALOG       32         301E2       %0>=       35       305C7       %MOD       32         301F6       %%0<=	300C7	%%MAX	34	304D5	%%SQRT	34
30112       %%0       35       30507       %%EXP       34         30123       %0       35       3051A       %EXP       32         30145       %%0=       35       3052D       %EXPM1       32         30156       %0=       35       30546       %kln       34         30173       %%0>       35       30559       %LN       32         30184       %0>       35       3056C       %LOG       32         301A6       %%0<	300E0	%MAX	33	304E1	%>%%SQRT	32
30123       %0        35       3051A       %EXP       32         30145       %%0 =       35       3052D       %EXPM1       32         30156       %0 =       35       30546       %kln       34         30173       %%0 >       35       30559       %LN       32         30184       %0 >       35       3056C       %LOG       32         301A6       %%0 <>/td>       35       3057F       %klnP1       34         301BA       %0 <>/td>       35       30592       %LNP1       32         301CE       %%0 <=	300F9	%MIN	33	304F4	%SQRT	32
30145       %%0=       35       3052D       %EXPM1       32         30156       %0=       35       30546       %kln       34         30173       %%0>       35       30559       %ln       32         30184       %0>       35       3056C       %LOG       32         301A6       %%0<	30112	%%O<	35	30507	%%EXP	34
30156       %0=       35       30546       %%LN       34         30173       %%0>       35       30559       %LN       32         30184       %0>       35       3056C       %LOG       32         301A6       %%0<	30123	%0<	35	3051A	%EXP	32
30156       %0=       35       30546       %%LN       34         30173       %%0>       35       30559       %LN       32         30184       %0>       35       3056C       %LOG       32         301A6       %%0<>       35       3057F       %%LNP1       34         301BA       %0<>       35,135       30592       %LNP1       32         301CE       %%0>=       35       305A5       %ALOG       32         301E2       %0>=       35       305C7       %MOD       32         301F6       %%0<=	30145	%%O=	35	3052D	%EXPM1	32
30184       %0>       35       3056C       %LOG       32         301A6       %%0<>/td>       35       3057F       %LNP1       34         301BA       %0<>/td>       35,135       30592       %LNP1       32         301CE       %%0>=       35       305A5       %ALOG       32         301E2       %0>=       35       305C7       %MOD       32         301F6       %%0<=	30156	%O=	35		%LN	34
301A6     %%0     35     3057F     %%LNP1     34       301BA     %0     35,135     30592     %LNP1     32       301CE     %%0>=     35     305A5     %ALOG     32       301E2     %0>=     35     305C7     %MOD     32       301F6     %%0<=	30173	%%O>	35	30559	LN	32
301BA     %0     35,135     30592     %LNP1     32       301CE     %%0>=     35     305A5     %ALOG     32       301E2     %0>=     35     305C7     %MOD     32       301F6     %%0<=	30184	%0>	35	3056C	%LOG	32
301CE       %%0>=       35       305A5       %ALOG       32         301E2       %0>=       35       305C7       %MOD       32         301F6       %%0<=	301A6	%%O<>	35	3057F	%%LNP1	34
301E2       %0>=       35       305C7       %MOD       32         301F6       %%0<=	301BA	%0<>	35, 135	30592	%LNP1	32
301F6       %%0<=	301CE	%%0>=	35	305A5	%ALOG	32
3020A     %     35     305F1     %%SIN     34       3025C     %     35     30602     %%SINDEG     34       3026A     %     35     30612     %%SINRAD     34       30275     %     35     3062B     %COS     32       30280     %     %     35     30642     %     %COS     34       3028B     %     %     35     30653     %     %COSDEG     34	301E2	%0>=	35	305C7	%MOD	32
3025C     %     35     30602     %%SINDEG     34       3026A     %%>     35     30612     %%SINRAD     34       30275     %>     35     3062B     %COS     32       30280     %%>=     35     30642     %%COS     34       3028B     %>=     35     30653     %%COSDEG     34	301F6	%%O<=	35	305DA	%SIN	32
3026A       %%>       35       30612       %%SINRAD       34         30275       %>       35       3062B       %COS       32         30280       %%>=       35       30642       %%COS       34         3028B       %>=       35       30653       %%COSDEG       34	3020A	%%<	35	305F1	%%SIN	34
30275       %>       35       3062B       %COS       32         30280       %%>=       35       30642       %*COS       34         3028B       %>=       35       30653       %*COSDEG       34	3025C	%<	35	30602	%%SINDEG	34
30280 %%>= 35 30642 %%COS 34 3028B %>= 35 30653 %%COSDEG 34	3026A	%%>	35	30612	%%SINRAD	34
3028B %>= 35 30653 %%COSDEG 34	30275	%>	35	3062B	%COS	32
3028B %>= 35 30653 %%COSDEG 34	30280	%%>=	35	30642	%%COS	34
30296		%>=		30653	%%COSDEG	
	30296	%%<=	35	30663	%%COSRAD	34

Addr.	Name	Page	Addr.	Name	Page
3067C	%TAN	32	31066	aMODF	
30693	%%TANRAD	34	31123	aH>HMS	
306AC	%ASIN	32	31219	$Y \le X$	
306C3	%%ASINRAD	34	3125D	TST15	
306DC	%ACOS	32	313D3	RNDC[B]	
306F3	%%ACOSRAD	34	314CA	GETAB1	
3070C	%ATAN	32	314E4	GETAB0	
30723	%ANGLE	32	31518	GETCD0	
3073A	%%ANGLE	34	31532	PUTAB0	
30746	%>%%ANGLE	32	31568	1/X15	
30757	%%ANGLEDEG	34	315BB	ADDF	
30767	%%ANGLERAD	34	317EE	SQRF	
30780	%%SINH	34	31994	DIV2	
30799	%SINH	32	319C1	CLRFRC	
307B2	%%COSH	34	33107	any	10
307C5	%COSH	32	33107	ZERO	10
307D8	%TANH	32	33107	BINT0	10
307EB	%ASINH	32	33111	real	10
307FE	%ACOSH	32	33111	MEMERR	10
30811	%ATANH	32	33111	ONE	10
30824	%EXPONENT	32	33111	BINT1	10
30837	%NFACT	33	3311B	cmp	10
3084D	%COMB	33	3311B	TWO	10
30860	%PERM	33	3311B	BINT2	10
30912	%%H>HMS	172	33125	THREE	10
30938	%FP	32	33125	str	10
3094B	%IP	32	33125	BINT3	10
3095E	%CEIL	32	3312F	BINT4	10
30971	%FLOOR	32	3312F	FOUR	10
30984	%%FLOOR	34	3312F	arry	10
30984	%%INT	34	33139	FIVE	10
309AD	%RAN	33	33139	list	10
30A2F	%RANDOMIZE	33	33139	BINT5	10
30A66	DORANDOMIZE	33	33143	id	10
30AAF	%FACT	33	33143	SIX	10
30BEA	887	30	33143	idnt	10
30CC7	%%12	30	33143	BINT6	10
30CEB	8860	30	3314D	SEVEN	10
30DC8	%%.4	30	3314D	BINT7	10
30E47	2%>%%	31	3314D	lam	10
30E5B	2%%>%	31	33157	seco	10
30E79	%REC>%POL	33	33157	BINT8	10
30E83	%%R>P	34	33157	EIGHT	10
30EA6	%POL>%REC	33	33161	NINE	10
30EB0	%%P>R	34	33161	symb	10
30EDD	%SPH>%REC	33	33161	BINT9	10
30F14	RNDXY	32	3316B	BINT10	10
30F28	TRCXY	32	3316B	sym	10

3067C - 3330F 577

Addr.	Name	Page	Addr.	Name	Page
3316B	TEN	10	33229	BINT29	11
33175	hxs	10	33233	THIRTY	12
33175	BINT11	10	33233	REALEXT	12
33175	ELEVEN	10	33233	BINT30	12
3317F	grob	11	3323D	THIRTYONE	12
3317F	TWELVE	11	3323D	BINT31	12
3317F	BINT12	11	33247	BINT32	12
33189	TAGGED	11	33247	THIRTYTWO	12
33189	BINT13	11	33251	THIRTYTHREE	12
33189	THIRTEEN	11	33251	BINT33	12
33193	FOURTEEN	11	3325B	THIRTYFOUR	12
33193	BINT14	11	3325B	BINT34	12
33193	EXT	11	33265	THIRTYFIVE	12
33193	unitob	11	33265	BINT35	12
3319D	FIFTEEN	11	3326F	TTHIRTYSIX	12
3319D	rompointer	11	3326F	BINT36	12
3319D	BINT15	11	33279	THIRTYSEVEN	12
331A7	SIXTEEN	11	33279	BINT37	12
331A7	REALOB	11	33283	THIRTYEIGHT	12
331A7	BINT16	11	33283	BINT38	12
331B1	2REAL	11	3328D	BINT39	12
331B1	REALREAL	11	3328D	THIRTYNINE	12
331B1	SEVENTEEN	11	33297	FORTY	12
331B1	BINT17	11	33297	FOURTY	12
331BB	BINT18	11	33297	BINT40	12
331BB	EIGHTEEN	11	332A1	BINT41	12
331C5	BINT19	11	332A1	FORTYONE	12
331C5	NINETEEN	11	332AB	FORTYTWO	12
331CF	BINT20	11	332AB	BINT42	12
331CF	TWENTY	11	332B5	FORTYTHREE	12
331D9	TWENTYONE	11	332B5	BINT43	12
331D9	BINT21	11	332BF	BINT44	12
331E3	BINT22	11	332BF	FORTYFOUR	12
331E3	TWENTYTWO	11	332C9	FORTYFIVE	12
331ED	BINT23	11	332C9	BINT45	12
331ED	TWENTYTHREE	11	332D3	BINT46	12
331F7	BINT24	11	332D3	FORTYSIX	12
331F7	TWENTYFOUR	11	332DD	FORTYSEVEN	12
33201	BINT25	11	332DD	BINT47	12
33201	TWENTYFIVE	11	332E7	FORTYEIGHT	12
3320B	TWENTYSIX	11	332E7	BINT48	12
3320B	REALSYM	11	332F1	FORTYNINE	12
3320B	BINT26	11	332F1	BINT49	12
33215	TWENTYSEVEN	11	332FB	FIFTY	13
33215	BINT27	11	332FB	BINT50	13
3321F	BINT28	11	33305	BINT51	13
3321F	TWENTYEIGHT	11	33305	FIFTYONE	13
33229	TWENTYNINE	11	3330F	FIFTYTWO	13

Addr.	Name	Page	Addr.	Name	Page
3330F	BINT52	13	33409	BINT77	14
33319	STRLIST	13	33413	BINT78	14
33319	THREEFIVE	13	3341D	BINT79	14
33319	FIFTYTHREE	13	3341D	SEVENTYNINE	14
33319	BINT53	13	33427	EIGHTY	14
33323	FIFTYFOUR	13	33427	BINT80	14
33323	BINT54	13	33431	LISTREAL	14
3332D	BINT55	13	33431	EIGHTYONE	14
3332D	FIFTYFIVE	13	33431	BINT81	14
33337	BINT56	13	3343B	BINT82	14
33337	FIFTYSIX	13	3343B	LISTCMP	14
33341	BINT57	13	33445	BINT83	14
33341	FIFTYSEVEN	13	33445	FIVETHREE	14
3334B	FIFTYEIGHT	13	3344F	BINT84	14
3334B	BINT58	13	3344F	FIVEFOUR	14
33355	FIFTYNINE	13	33459	BINT85	14
33355	BINT59	13	33459	2LIST	14
3335F	BINT60	13	33463	FIVESIX	14
3335F	SIXTY	13	33463	BINT86	14
33369	BINT61	13	3346D	LISTLAM	14
33369	SIXTYONE	13	3346D	BINT87	14
33373	BINT62	13	33477	BINT88	14
33373	SIXTYTWO	13	33481	BINT89	14
3337D	SIXTYTHREE	13	3348B	BINT90	14
3337D	BINT63	13	33495	BINT91	14
33387	YHI	13	33495	BINT_91d	14
33387	SIXTYFOUR	13	3349F	BINT92	14
33387	BINT64	13	334A9	BINT93	14
33387	BINT40h	13	334B3	BINT94	15
33391	ARRYREAL	13	334BD	BINT95	15
33391	BINT65	13	334C7	BINT_96d	15
3339B	FOURTWO	13	334C7	BINT96	15
3339B	BINT66	13	334D1	BINT97	15
333A5	BINT67	13	334D1	IDREAL	15
333A5	FOURTHREE	13	334DB	BINT98	15
333AF	BINT68	14	334E5	BINT99	15
333AF	SIXTYEIGHT	14	334EF	BINT100	15
333B9	FOURFIVE	14	334EF	ONEHUNDRED	15
333B9	BINT69	14	334F9	BINT101	15
333C3	BINT70	14	33503	BINT102	15
333C3	SEVENTY	14	3350D	BINT103	15
333CD	BINT71	14	33517	BINT104	15
333D7	BINT72	14	33521	BINT105	15
333E1	BINT73	14	3352B	BINT106	15
333EB	BINT74	14	33535	BINT107	15
333EB	SEVENTYFOUR	14	3353F	BINT108	15
333F5	BINT75	14	33549	BINT109	15
333FF	BINT76	14	33553	BINT110	15

3330F - 338A5 579

Addr.	Name	Page	Addr.	Name	Page
3355D	char	15	336CF	2HXS	16
3355D	BINT111	15	336D9	BINTC0h	16
33567	BINT112	15	336E3	2GROB	16
33571	BINT113	15	336ED	TAGGEDANY	16
3357B	BINT114	15	336F7	EXTREAL	16
33585	BINT115	15	33701	EXTSYM	16
33585	BINT_115d	15	3370B	2EXT	16
3358F	BINT116	15	33715	ROMPANY	16
3358F	BINT_116d	15	3371F	BINT253	16
33599	BINT117	15	33729	BINT255d	17
335A3	BINT118	15	33733	REALOBOB	17
335AD	BINT119	15	3373D	#_102	17
335B7	BINT120	15	33747	#SyntaxErr	17
335C1	BINT121	15	33751	BINT_263d	17
335CB	BINT122	15	3375B	#110	17
335CB	BINT_122d	15	33765	3REAL	17
335D5	BINT123	15	3376F	Err#Kill	17
335DF	BINT124	15	33779	Err#NoLstStk	17
335E9	BINT125	15	33783	#NoRoomForSt	17
335F3	BINT126	16	3378D	#132	17
335FD	BINT127	16	33797	REALSTRSTR	17
33607	BINT128	16	337A1	#134	17
33607	BINT80h	16	337AB	#135	17
33611	BINT129	16	337B5	#136	17
3361B	BINT130	16	337BF	#137	17
3361B	BINT130d	16	337C9	#138	17
3361B	BINT_130d	16	337D3	#139	17
3361B	XHI-1	16	337DD	#13A	17
33625	XHI	16	337E7	#13B	17
33625	BINT_131d	16	337F1	#13D	17
33625	BINT131d	16	337FB	#13E	17
33625	BINT131	16	33805	INTEGER337	17
3362F	#8F	16	3380F	#200	17
33639	SYMBREAL	16	33819	Err#NoLstArg	17
33643	#92	16	33823	STRREALREAL	17
3364D	#9A	16	3382D	ARRYREALREAL	17
33657	SYMBUNIT	16	33837	#412	17
33661	backup		33841	#444	17
3366B	SYMOB	16	3384B	ARRYLISTREAL	18
33675	SYMREAL	16	33855	#452	18
3367F	#A2	16	3385F	#510	18
33689	#A5	16	33869	#511	18
33693	SYMID	16	33873	#550	18
3369D	SYMLAM	16	3387D	IDREALOB	18
336A7	#A9	16	33887	IDLISTOB	19
336B1	SYMSYM	16	33891	#700	19
336BB	SYMEXT	16	3389B	#861	19
336C5	HXSREAL	16	338A5	#862	19

Addr.	Name	Page	Addr.	Name	Page
338AF	#865	19	33B07	tokWHERE	45
338B9	#86E	19	33B13	14SPACES\$	45
338C3	ATTNERR	19	33B39	NEWLINE\$	43
338CD	#A11	19	33B45	\$DER	45
338D7	#A12	19	33B55	tok_	43
338E1	#A1A	19	33B55	SPACE\$	43
338EB	#A21	19	33B61	tokUNKNOWN	45
338F5	#A22	20	33B79	tokquote	45
338FF	#A2A	20	33B85	tok'	44
33909	#A61	20	33B91	tok,	44
33913	#A62	20	33B9D	tok.	44
3391D	#A65	20	33BA9	tok;	45
33927	#A6E	20	33BB5	toklparen	45
33931	#AA1	20	33BC1	tokrparen	45
3393B	#AA2	20	33BCD	tok^	45
33945	#AAA	20	33BD9	tok*	45
3394F	#C06	20	33BE5	tok/	45
33959	#C07	20	33BF1	tok+	45
33963	#C08	20	33BFD	tok-	44
3396D	Connecting	20	33C09	tok=	44
33977	#C0B	20	33C15	tokSQRT	45
33981	#CAlarmErr	20	33C21	tokDER	45
3398B	EXTOBOB	20	33C2D	tokCTGROB	45
33995	#EXITERR	21	33C3F	tokCTSTR	45
3399F	MINUSONE	$\frac{-1}{21}$	33C4D	tok0	44
339A9	%e	28	33C59	tok1	44
339BE	%.5	28	33C65	tok2	45
339D3	%- <b>.</b> 5	28	33C71	tok3	45
339E8	%10	28	33C7D	tok4	45
339FD	%180	29	33C89	tok5	45
33A12	%200	29	33C95	tok6	45
33A27	%360	29	33CA1	tok7	45
33A3C	%400	29	33CAD	tok8	44
33A51	tok]	45	33CB9	tok9	44
33A5D	lbrac		33D2B	CHR_00	41
33A6B	tok[	45	33D32	CHR	41
33A77	tok{	44	33D39	_ CHR_DblQuote	41
33A83	tok}	45	33D40	CHR_#	41
33A8F	toksharp	45	33D47	CHR_*	41
33A9B	tokuscore	45	33D4E	CHR_+	41
33AA7	tok\$	45	33D55	CHR_,	41
33AB3	tok&	45	33D5C	CHR	41
33ABF	tokESC	44	33D63	CHR	41
33ACB	tok>>	45	33D6A	CHR_/	41
33AD7	tok<<	44	33D71	CHR_0	41
33AE3	tokexponent	44	33D78	CHR_1	41
33AEF	tokanglesign	45	33D7F	CHR_2	41
33AFB	tokSIGMA	45	33D86	CHR_3	41

338AF - 34088 581

Addr.	Name	Page	Addr.	Name	Page
33D8D	CHR_4	41	33EDD	CHR_1	42
33D94	CHR_5	41	33EE4	CHR_m	43
33D9B	CHR_6	41	33EEB	CHR_n	43
33DA2	CHR_7	41	33EF2	CHR_o	43
33DA9	CHR_8	41	33EF9	CHR_p	43
33DB0	CHR_9	41	33F00	CHR_q	43
33DB7	CHR_:	41	33F07	CHR_r	43
33DBE	CHR_;	41	33F0E	CHR_s	43
33DC5	CHR_<	41	33F15	CHR_t	43
33DCC	CHR_=	41	33F1C	CHR_u	43
33DD3	CHR_>	41	33F23	CHR_v	43
33DDA	CHR_A	41	33F2A	CHR_w	43
33DE1	CHR_B	41	33F31	CHR_x	43
33DE8	CHR_C	41	33F38	CHR_y	43
33DEF	CHR_D	42	33F3F	CHR_z	43
33DF6	CHR_E	42	33F46	CHR>	43
33DFD	CHR_F	42	33F4D	CHR_<<	43
33E04	CHR_G	42	33F54	CHR_>>	43
33E0B	CHR_H	42	33F5B	CHR_Angle	43
33E12	CHR_I	42	33F62	CHR_Deriv	43
33E19	CHR_J	42	33F69	CHR_Integral	43
33E20	CHR_K	42	33F70	CHR_LeftPar	41
33E27	CHR_L	42	33F77	CHR_Newline	41
33E2E	CHR_M	42	33F7E	CHR_Pi	43
33E35	CHR_N	42	33F85	CHR_RightPar	41
33E3C	CHR_O	42	33F8C	CHR_Sigma	43
33E43	CHR_P	42	33F93	CHR_Space	41
33E4A	CHR_Q	42	33F9A	CHR_UndScore	42
33E51	CHR_R	42	33FA1	CHR_[	42
33E58	CHR_S	42	33FA8	CHR_]	42
33E5F	CHR_T	42	33FAF	CHR_{	43
33E66	CHR_U	42	33FB6	CHR_}	43
33E6D	CHR_V	42	33FBD	CHR_<=	43
33E74	CHR_W	42	33FC4	CHR_>=	43
33E7B	CHR_X	42	33FCB	CHR_<>	43
33E82	CHR_Y	42	33FD2	\$_R<<	44
33E89	CHR_Z	42	33FE2	\$_R <z< td=""><td>44</td></z<>	44
33E90	CHR_a	42	33FF2	\$_XYZ	44
33E97	CHR_b	42	34002	\$_<<>>	44
33E9E	CHR_c	42	34010	<b>\$_</b> {}	44
33EA5	CHR_d	42	3401E	\$_[]	44
33EAC	CHR_e	42	3402C	\$_''	44
33EB3	CHR_f	42	3403A	\$_::	44
33EBA	CHR_g	42	34048	<pre>\$_LRParens</pre>	44
33EC1	CHR_h	42	34056	\$_2DQ	44
33EC8	CHR_i	42	34064	\$_ECHO	44
33ECF	CHR_j	42	34076	\$_EXIT	44
33ED6	CHR_k	42	34088	<pre>\$_Undefined</pre>	44

Addr.	Name	Page	Addr.	Name	Page
340A4	\$_RAD	44	343E1	2RDROP	129
340B4	\$_GRAD	44	343F3	3RDROP	129
34144	RSWAP	129	34405	#-PICK	110
3416E	XYZ>YXZ	107	34417	#+PICK	110
3416E	ROTSWAP	107	34431	DUP#1+PICK	77, 106
34195	XYZ>ZY	107	34436	#1+PICK	109
34195	ROTDROPSWAP	107	34451	#2+PICK	109
341A8	XYZ>YZ	107	34465	#3+PICK	109
341A8	ROTDROP	107	34474	#4+PICK	110
341BA	UNROTSWAP	107, 108	34485	3PICK	109
341BA	XYZ>ZYX	107, 108	3448A	4PICK	109
341BA	SWAPROT	107, 108	3448F	5PICK	109
341D2	3DROP	106	34494	6PICK	109
341D2	XYZ>	106	34499	7PICK	109
341D7	XYZW>	106	3449E	8PICK	109
341D7	4DROP	106	344A3	9PICK	109
341DC	5DROP	106	344A8	10PICK	109
341E8	6DROP	106	344CB	#-ROLL	108
341F4	7DROP	106	344DD	#+ROLL	108
3421A	SWAPDROP	107	344F2	#1+ROLL	108
3421A	XY > Y	107	34504	get1	77
3422B	3UNROLL	108, 108	34517	#2+ROLL	108
3422B	UNROT	108, 108	3452B	#-UNROLL	109
3422B	XYZ>ZXY	108, 108	3453D	#+UNROLL	109
3423A	XYZW>YZWX	107	34552	#1+UNROLL	109
3423A	FOURROLL	107	34564	#2+UNROLL	109
3423A	4ROLL	107	3457F	DUPUNROT	106, 107
34257	5ROLL	108	3457F	SWAPOVER	106, 107
34257	FIVEROLL	108	34611	1PUTLAM	118
34281	6ROLL	108	34616	1GETLAM	117
34281	SIXROLL	108	3461B	2PUTLAM	118
342BB	EIGHTROLL	108	34620	2GETLAM	117
342BB	8ROLL	108	34625	3PUTLAM	118
342EA	SEVENROLL	108	3462A	3GETLAM	117
342EA	7ROLL	108	3462F	4PUTLAM	118
3432C	DUP4UNROLL	106	34634	4GETLAM	117
34331	FOURUNROLL	108	34639	5PUTLAM	118
34331	4UNROLL	108	3463E	5GETLAM	117
34331	XYZW>WXYZ	108	34643	6PUTLAM	118
34357	5UNROLL	109	34648	6GETLAM	117
34357	FIVEUNROLL	109	3464D	7PUTLAM	118
3438D	SIXUNROLL	109	34652	7GETLAM	117
3438D	6UNROLL	109	34657	8PUTLAM	118
343BD	XYZ>Z	107, 108	3465C	8GETLAM	117
343BD	UNROT2DROP	107, 108	34661	9PUTLAM	118
343BD	ROTROT2DROP	107, 108	34666	9GETLAM	117
343CF	4UNROLL3DROP	108	3466B	10PUTLAM	118
343CF	XYZW>W	108	34670	10GETLAM	117

340A4 - 350C3 583

Addr.	Name	Page	Addr.	Name	Page
34675	11PUTLAM	118	34999	EQcase	144
3467A	11GETLAM	118	349B1	caseDROP	139
3467F	12PUTLAM	118	349C6	NOTcaseDROP	139
34684	12GETLAM	118	349D6	case2DROP	140
34689	13PUTLAM	118	349EA	NOTcase2DROP	140
3468E	13GETLAM	118	349F9	case	139
34693	14PUTLAM	118	34A13	NOTcase	139
34698	14GETLAM	118	34A22	IT	138
3469D	15PUTLAM	118	34A31	GOTO	129
346A2	15GETLAM	118	34A46	?GOTO	129
346A7	16PUTLAM	118	34A59	NOT?GOTO	129
346AC	16GETLAM	118	34A7E	#0=?SEMI	141
346B1	17PUTLAM	118	34A92	NOT?SEMI	137
346B6	17GETLAM	118	34AA1	?SEMI	137
346BB	18PUTLAM	118	34AAD	SEMILOOP	151
346C0	18GETLAM	118	34ABE	ITE_DROP	139
346C5	19PUTLAM	118	34AD3	COLA_EVAL	131
346CA	19GETLAM	118	34AF4	COLARPITE	138
346CF	20PUTLAM	118	34B3E	ITE	139
346D4	20GETLAM	118	34B4F	2'RCOLARPITE	138
346D9	21PUTLAM	119	34BAB	2@REVAL	128
346DE	21GETLAM	118	34BBB	3@REVAL	128
346E3	22PUTLAM	119	34BD8	NOT?DROP	138
346E8	22GETLAM	118	34BEF	ticR	128
346ED	23PUTLAM	119	34C82	EXPAND	48, 57
346F2	23GETLAM	118	34D00	CACHE	117
346F7	24PUTLAM	119	34D51	SAVELAM	
346FC	24GETLAM	118	34D58	SAVESTACK	117
34701	25PUTLAM	119	34EBE	DUMP	117
34706	25GETLAM	118	34FA6	undo	117
3470B	26PUTLAM	119	34FC0	DUPROM-WORD?	100
34710	26GETLAM	118	34FCD	ROM-WORD?	100
34715	27PUTLAM	119	34FE6	Rom-Word?	
3471A	27GETLAM	118	35018	2SWAP	107
34797	DUP4PUTLAM	119	35037	DUPTYPECHAR?	200
347AB	DUPTEMPENV	119	3503C	TYPECHAR?	200
3483E	GETLAMPAIR	119	35046	DUPTYPEIDNT?	199
348D2	#=case	141	3504B	TYPEIDNT?	199
348E2	OVER#=case	141	35082	DUPTYPEFLASHPTR?	200
348F7	DUP#0=case	141	35087	TYPEFLASHPTR?	200
348FC	#0=case	141	35091	DUPTYPEZINT?	200
3490E	DUP#0=csedrp	141	35096	TYPEZINT?	200
34920	EQcasedrop	144	350A0	DUPTYPELNGREAL?	200
34939	#=casedrop	141	350A5	TYPELNGREAL?	200
3494E	NOTcasedrop	139	350AF	DUPTYPELNGCMP?	200
3495D	casedrop	139	350B4	TYPELNGCMP?	200
34976	NOTcase2drop	139	350BE	DUPTYPEFONT?	200
34985	case2drop	139	350C3	TYPEFONT?	200

Addr.	Name	Page	Addr.	Name	Page
350CD	DUPTYPEAPLET?	200	3530D	#1<>	25
350D2	TYPEAPLET?	200	3531C	DUP#1=	26
350DC	DUPTYPELAM?	199	3532B	DUP#0<>	26
350E1	TYPELAM?	199	3533C	!insert\$	49
350EB	DUPTYPEBINT?	200	35346	SWAP&\$	49
350F0	TYPEBINT?	200	35369	!!append\$?	49
350F5	#37258	21	353CD	!append\$	49
350FA	DUPTYPEHSTR?	199	353EB	!!insert\$	49
350FF	TYPEHSTR?	199	353F7	!!append\$	49
35109	DTYPECSTR?	199	354CB	'RSaveRomWrd	197
35109	DUPTYPECSTR?	199	354CB	'RSAVEWORD	197
3510E	TYPECSTR?	199	35511	#MIN	25
35118	DTYPEREAL?	199	3551D	#MAX	25
35118	DUPTYPEREAL?	199	35552	#-#2/	24
3511D	TYPEREAL?	199	3558C	DROPZERO	21
35127	DUPTYPECMP?	199	355A5	2DROP00	21
3512C	TYPECMP?	199	355D0	#6-	23
35136	DUPTYPEARRY?	199	355D5	#5-	23
35136	DTYPEARRY?	199	355DA	#4-	23
3513B	TYPEARRY?	199	355DF	#3-	23
35145	DUPTYPEROMP?	200	355FD	#3+	23
3514A	TYPEROMP?	200	35602	#4+	23
35154	DUPTYPERRP?	200	35607	#5+	23
35159	TYPERRP?	200	3560C	#6+	23
35163	DUPTYPESYMB?	199	35611	#7+	23
35168	TYPESYMB?	199	35616	#8+	23
35172	DTYPECOL?	200	3561B	#9+	23
35172	DUPTYPECOL?	200	35620	#10+	23
35177	TYPECOL?	200	3562A	#12+	23
35181	DUPTYPEGROB?	199	35675	#10*	23
35186	TYPEGROB?	199	3569B	#8*	23
35190	DUPTYPELIST?	199	356B8	#6*	23
35190	DTYPELIST?	199	356D5	5skipcola	131
35195	TYPELIST?	199	35703	3skipcola	131
3519F	DUPTYPETAG?	200	3570C	2skipcola	131
351A4	TYPETAGGED?	200	35715	skipcola	131
351AE	DUPTYPEEXT?	200	3571E	DUP#2+	24
351B3	TYPEEXT?	200	35733	DROPSWAP	106
351FA	OverWrF/TLp		3574D	XYZ>Y	106, 107
35268	OVER#=	25	3574D	ROT2DROP	106, 107
35280	DROPTRUE	136	3574D	DROPSWAPDROP	106, 107
35289	DROPFALSE	136	3576E	SWAPDUP	107
35292	TYPERARRY?	199	3579C	ROTDUP	107
352AD	TYPECARRY?	199	357BB	SWAP#-	24
352BD	DUP#0=	25	357CE	DROPDUP	106
352E0	#3=	25	357E2	DUPLEN\$	47
352F1	#2=	25	357FC	#+DUP	24
352FE	#1=	25	3581F	#-DUP	24

350CD - 36043 585

Addr.	Name	Page	Addr.	Name	Page
35830	#1+DUP	24	35CE0	DUPDUP	106
35841	#1-DUP	24	35CF4	OVERDUP	109
35857	SWAPDROPDUP	107	35D08	COERCEDUP	22
35872	SWAPDROPSWAP	107, 108	35D1C	UNROTDUP	108
35872	XYZ>ZX	107, 108	35D30	2DUPSWAP	106, 106
35872	UNROTDROP	107, 108	35D30	DUP3PICK	106, 106
3588B	4ROLLDROP	107	35D44	4UNROLLDUP	108
358A7	5ROLLDROP	108	35D58	NTHCOMDDUP	69
358C2	2DUP#<	25	35D6C	OVERSWAP	109, 109
358DC	2DUP#=	25	35D6C	OVERUNROT	109, 109
358F8	2DUP#>	25	35D80	ROLLSWAP	108
35912	DUP#1+	24	35D94	NULL\$SWAP	46
3592B	SWAP#1+	24,77	35DA8	SUB\$SWAP	48
3592B	SWP1+	24,77	35DBC	%MAXorder	33
35956	DUP#1-	24	35DDA	?SKIPSWAP	138
3596D	DROPONE	21	35DEE	1ABNDSWAP	119
3597F	RDROPCOLA	129	35E07	ROT+SWAP	25
35994	COLACOLA	131	35E07	ROT#+SWAP	25
359AD	COLAcase	140	35E20	4PICK#+SWAP	25
359C8	COLANOTcase	140	35E20	4PICK+SWAP	25
359E3	ORcase	139	35E39	#+SWAP	24
359F7	REQcase	144	35E4D	#-SWAP	24
35A10	REQcasedrop	144	35E61	#1+SWAP	24
35A29	SAFESTO	167	35E75	ZEROSWAP	22
35A56	DUPSAFE@	166	35E89	#1-1SWAP	24
35A5B	SAFE@	166	35EA2	ONESWAP	22
35A88	?>ROMPTR	100	35EB6	COERCESWAP	22
35AAB	?ROMPTR>	100	35ECA	%>%%SWAP	31
35AE2	MACRODCMP		35EDE	%%*SWAP	33
35B32	2DROPFALSE	136	35EF2	XYZ>ZTRUE	136
35B46	PALPTRDCMP		35F06	4ROLLSWAP	107
35B82	palrompdcmp	55	35F1A	3PICKSWAP	109
35B96	#0=UNTIL	151	35F2E	4PICKSWAP	109
35BAF	INCOMPDROP	71	35F42	1GETSWAP	119
35BC3	NTHCOMPDROP	69	35F56	?SWAP	138
35BD7	APPEND_SPACE	49	35F6A	!append\$SWAP	49
35BEB	7UNROLL	109	35F7E	NOT?SWAPDROP	138
35BFF	RESOROMP	100	35F97	?SWAPDROP	138
35C18	%10*	31	35FB0	N+1DROP	75, 106
35C2C	DUP@	166	35FB0	#1+NDROP	75, 106
35C40	DUPROMPTR@	100	35FC4	ROLLDROP	108
35C54	#=ITE	141	35FD8	MDIMSDROP	64
35C68	INNERDUP	71	35FF3	DUPROT	106
35C7C	NOTAND	136	36007	DROPROT	106
35C90	TOTEMPSWAP	171	3601B	#1-ROT	24
35CA4	ROT2DUP	107	3602F	%%*ROT	33
35CB8	ROTAND	136	36043	FOURROLLROT	108
35CCC	ROTOVER	107	36043	4ROLLROT	108

Addr.	Name	Page	Addr.	Name	Page
36057	4UNROLLROT	108	3640F	NOT_UNTIL	151
3606B	DROPOVER	106	36428	NOT_WHILE	151
3607F	EQOVER	137	36441	DUP#0<>WHILE	151
36093	#+OVER	24	3645A	DUPINDEX@	152
360A7	#-OVER	24	3646E	SWAPINDEX@	152
360BB	ZEROOVER	22	36482	OVERINDEX@	152
360CF	UNROTOVER	108	36496	SWAPLOOP	151
360E3	4ROLLOVER	108	364AF	DROPLOOP	151
360F7	3PICKOVER	109	364C8	DUP#0_DO	151
3610B	4PICKOVER	109	364E1	toLEN_DO	151
3611F	DUPPICK	106	364FF	1GETABND	119
36133	DUPROLL	106	36513	DUP1LAMBIND	116
36147	OVER#2+UNROL	77	36518	1LAMBIND	116
3615B	8UNROLL	109	3652C	caseTRUE	140
3616F	10UNROLL	109	36540	TrueFalse	136
36183	OVERARSIZE	64	36540	TRUEFALSE	136
3619E	'ERRJMP	130	36554	FalseTrue	136
361B2	caseERRJMP	146	36554	FALSETRUE	136
361C6	?CARCOMP	68	36568	ZEROFALSE	22
361DA	NEWLINE\$&\$	47	3657C	ONEFALSE	22
361DA	NEWLINE&\$	47	36590	#=casedrpfls	141
361EE	#1-{}N	72	365B3	casedrpfls	139
36202	TWO{}N	72	365CC	case2drpfls	140
36216	THREE { } N	72	365E5	caseFALSE	140
3622A	DUPINCOMP	71	365F9	ORNOT	136
3623E	SWAPINCOMP	71	3660D	EQUALNOT	137
36252	DUPNULL\$?	55	36621	2DUPEQ	137
36266	DUPNULLCOMP?	68	36635	DUPEQ:	137
3627A	DUPLENCOMP	68	3663A	EQ:	137
3628E	#1-SUB\$	48	3664E	EQOR	137
362A2	1_#1-SUB\$	48	36662	EQUALOR	137
362A2	1_#1-SUB	48	36676	2#0=OR	26
362B6	LAST\$	48	36694	OVER#0=	25
362CA	#1+LAST\$	48	366A8	OVER#<	25
362DE	DUP\$>ID	116	366BC	#<3	25
362F2	SWAP%>C%	37	366D0	DUP#<7	26
36306	'NOP	130	366E9	INNER#1=	71
3631A	::NEVAL	74	366FD	#5=	25
3632E	2GETEVAL	119	36711	#2<>	25
36342	DROPRDROP	129	36725	OVER#>	26
3635B	SWAPCOLA	131	36739	ONE#>	25
3636F	XYZ>ZCOLA	131	36739	#>1	25
36383	#0=?SKIP	141	3674D	DUP3PICK#+	$\frac{26}{24}$
3639C	#1=?SKIP	142	3674D	2DUP#+	24
363B5	#=?SKIP	140	36761	ROT#+	24
363CE	ONE_EQ	25	36775	OVER#+	24
363E2	#>?SKIP	140	36789	3PICK#+	25
363FB	COLASKIP	131	3679D	4PICK#+	25 25
20211	COLLIDITE	101	30,70	11 101011	20

36057 - 36F79 587

Addr.	Name	Page	Addr.	Name	Page
367B1	ROT#-	24	36BAA	NcaseTYPEERR	146
367C5	OVER#-	24	36BBE	'x*	130
367D9	INDEX@#-	152	36BD2	'xDER	130
367ED	SWAPOVER#-	24	36BE6	%%/ <b>&gt;</b> %	34
36801	ROT#1+	24	36BFA	UNCOERCE%%	31
36815	#-+1	23	36C0E	DUP%0=	35
36815	#1-	23	36C22	SWAP%%/	34
36829	SWAP#1-	24	36C36	caseDrpBadKy	146
3683D	DROP#1-	24	36C4F	caseDEADKEY	146
36851	#1-+	23	36C4F	caseDoBadKey	146
36851	#+-1	23	36C68	GROBDIMw	90
36851	\$1-+	23	36C7C	%%*UNROT	33
36865	COLAITE	139	36C90	XYZW>YWZX	107
36883	ERROROUT	156	36C90	SWAP4ROLL	107
368B5	SWAP2DUP	107	36CA4	2DUP5ROLL	106
368C9	RSKIP	129	36CB8	SWAP3PICK	107
368E7	GROB!ZERODRP	91	36CCC	3PICK3PICK	109
368FB	casedrptru	139	36CE0	SWAP4PICK	109
36914	NOTcaseTRUE	140	36CF4	OVER5PICK	109
3692D	?SEMIDROP	138	36D08	EQUALcasedrp	144
36946	SWAPUnDROP	77	36D21	DUP#0=csDROP	142
3695A	SWAPUnNDROP	76	36D3A	jEQcase	144
3696E	DUP'	130	36D4E	ANDcase	139
36982	SWAP'	130	36D62	EQUALcase	144
36996	DROP'	130	36D76	# <case< td=""><td>141</td></case<>	141
369AA	OVER'	130	36D8A	#1=case	142
369BE	STO'	130	36D9E	#<>case	141
369D2	TRUE'	130	36DB2	#>2case	142
369E6	ONEFALSE'	130	36DCB	#>case	141
369FF	FALSE'	130	36DDF	j%0=case	142
36A13	#1+'	130	36DF3	REALcase	145
36A27	'R'R	128	36E07	dARRYcase	145
36A4A	'RRDROP	128	36E43	dLISTcase	145
36A63	ONECOLA	131	36E57	EditExstCase	146
36A77	dvarlsBIND	117	36E6B	ANDNOTcase	139
36A8B	'LAMLNAMESTO	168	36E7F	EQUALNOTcase	144
36AA4	'xDEREQ	131	36E93	dIDNTNcase	145
36ABD	DUPNULL{}?	72	36EA7	dREALNcase	145
36AD6	DUPZERO	21	36EBB	EOIT	144
36AEA	DUPONE	$\frac{-1}{21}$	36ED4	DUP#0=IT	141
36AFE	SWAPONE	$\frac{-1}{21}$	36EED	ANDITE	139
36B12	ONEONE	$\frac{-1}{21}$	36F01	EQITE	144
36B12	ONEDUP	21	36F15	#0=ITE	141
36B26	DUPTWO	21	36F29	# <ite< td=""><td>141</td></ite<>	141
36B3A	NOTcsdrpfls	139	36F3D	#>ITE	141
36B53	caseSIZEERR	146	36F51	DUP#0=ITE	141
36B67	NcaseSIZEERR	146	36F65	UserITE	146
36B7B	CKREAL	198	36F79	SysITE	146
עועוטנ	CICICII	100	1 301 / 3	575111	140

Addr.	Name	Page	Addr.	Name	Page
36F8D	top&Cr		37784	NTHOF	69
36FA6	metaROTDUP	76	37798	Find1stTrue	70
36FBA	ROTUntop&	76	377C5	Lookup	70
36FCE	roll2top&	76	377DE	Lookup.1	70
36FCE	rolltwotop&	76	37829	EQLookup	70
36FE2	plDRPpZparg	88	378FA	POS\$	47
36FF6	&\$SWAP	49	378FA	POSCHR	47
3700A	SWAPCKREF	171	37906	POSCHRREV	47
3701E	pZpargSWAPUn	88	37906	POS\$REV	47
37032	DROPNDROP	75, 106	37AA5	CHR>\$	47
37046	20VER	109	37ABE	STRIPTAGS	61
3705A	?Ob>Seco	73	37AEB	STRIPTAGS12	62
37073	Ob>Seco	73	37B04	TAGOBS	61
37087	20b>Seco	73	37B54	NEXTCOMPOB	70
3709B	ZEROISTOPSTO	152	37C06	>LASTRAM-WORD	
3709B	ExitAtLOOP	152	37F48	xIF	462
370AF	RclHiddenVar	170	37F7F	xTHEN	473
370C3	WithHidden	170	3805D	xELSE	458
37104	StoHiddenVar	170	3807D	xIFEND	
37118	PuHiddenVar	170	38093	xALG->	
3712C	SaveVarRes	169	380DB	xWHILE	475
3714A	SetHiddenRes	170	38105	xREPEAT	469
37186	RestVarRes	169	3816B	xDO	457
371B3	Embedded?	69	38195	xUNTIL	474
371F9	UStackDepth		381AB	xSTART	471
3721C	Sig?ErrJmp	158	38252	xSTARTVAR	459
37258	DupAndThen		38266	#FFFF	21
37287	ZEROZERO	21	38275	#BB	16
37294	#ZERO#ONE	21	3831C	xNEXT	465
37305	#ZERO#SEVEN	21	3851F	xSTEP	472
37315	#ONE#27	21	387AC	xIFERR	462
37328	#TWO#ONE	21	3880D	xHALT	461
3733A	#TWO#TWO	21	38837	xSILENT'	
3734A	#TWO#FOUR	21	3885C	xRPN->	465
3735C	#THREE#FOUR	21	38999	x>>ABND	
3736E	#FIVE#FOUR	21	389B9	x<<	
37380	ZEROZEROZERO	21	389D4	x>>	
37394	ZEROZEROONE	21	389EF	x'	
373A8	ZEROZEROTWO	21	38A14	xENDTIC	
374BE	SWAPDROPTRUE	136	38A2F	xWHILEEND	
3760D	SubMetaOb	77	38A54	xENDDO	458
37685	SubMetaOb1	77	38ABA	xERRTHEN	
376B7	matchob?	69	38B28	xCASE	455
376C1	matchob?Lp		38B43	xTHENCASE	
376EE	POSCOMP	69	38BAE	xDIR	
37702	nextpos		38BBF	xPROMPT	467
37752	#=POSCOMP	69	38C00	DISPST2&FREEZE	282
3776B	EQUALPOSCOMP	69	38C1B	xGROB	

36F8D - 3A57C 589

Addr.	Name	Page	Addr.	Name	Page
38C2C	xEVAL>		394F1	xKILL	463
38D2F	xNOEVAL>		3950C	xOFF	465
38D72	xSTRUCT>		39527	xDOERR	457
38D83	x <struct< td=""><td></td><td>3955B</td><td>xERR0</td><td>458</td></struct<>		3955B	xERR0	458
38D94	xSTRUCT->		39576	xERRN	458
38DE1	xASR	454	39591	xERRM	458
38E01	xRL	469	395AC	xEVAL	458
38E21	xRLB	469	395F3	XIFTE	462
38E41	xRR	470	396A4	xIFT	462
38E61	xRRB	470	39705	xSYSEVAL	473
38E81	xSL	471	39725	xDISP	457
38EA1	xSLB	471	39745	xFREEZE	460
38EC1	xSR	471	39765	xBEEP	454
38EE1	xSRB	471	39785	x>NUM	465
38F01	xR>B	468	397E5	xLAST	463
38F21	xB>R	454	39819	TIAWx	475
38F41	xCONVERT	456	39839	xCLLCD	455
38F81	xUVAL	474	39854	xKEY	463
38FB5	x>UNIT	474	3989C	xCONT	456
38FD7	XUBASE	474	398B9	x=	476
3900B	XUFACT	474	39976	xNEG	465
3905D	xTIME	473	39A07	xABS	453
39078	xDATE	456	39A6C	xCONJ	456
39093	xTICKS	473	39AC7	xPI	466
390AE	xWSLOG	475	39AE4	xMAXR	464
390C9	xACKALL	453	39B01	xMINR	465
390E4	xACK	453	39B1E	xCONSTANTe	458
39104	xSETDATE	456	39B3B	xi	462
39124	xSETTIME	473	39B58	X+	476
39144	xCLKADJ	455	39C8B	SWAP>HCOMP	68
39164	xSTOALARM	472	39C9F	\$&ob	53
3918E	xRCLALARM	468	39CB3	ob&\$	53
391AE	xFINDALARM	459	39CD5	XNEGNEG	
391D8	xDELALARM	457	39CFC	X-	476
391F8	xTSTR	474	39DE8	x*	476
39218	xDDAYS	456	39E6B	#A4	16
39238	xDATE+	456	39F49	x/	476
39277	#B437D	21	3A097	x^	477
39332	?GETMSG	157	3A12D	#4FF	18
393CA	xCRDIR	456	3A1C2	#304	17
393EA	xPATH	466	3A200	rpnXROOT	
39405	XHOME	461	3A278	xXROOT	476
39420	xUPDIR	474	3A32B	XINV	462
3943B	xVARS	475	3A390	xARG	453
39456	xTVARS	474	3A3EE	xSIGN	471
39480	XBYTES	455	3A442	xSQRT	475
394AA	xNEWOB	465	3A4EF	xSQ	471
394C8	INHARDROM?	179	3A57C	xSIN	471

Addr.	Name	Page	Addr.	Name	Page
3A5D0	xCOS	456	3B477	xPERM	466
3A624	xTAN	473	3B4C9	xSF	471
3A678	xSINH	471	3B4E9	xCF	455
3A6C2	xCOSH	456	3B509	xFS?	460
3A70C	xTANH	473	3B529	xFC?	459
3A756	xASIN	454	3B549	xDEG	456
3A7DC	xACOS	453	3B564	xRAD	468
3A844	xATAN	454	3B57F	xGRAD	460
3A88E	xASINH	454	3B59A	xFIX	459
3A8D8	xACOSH	453	3B5BA	xSCI	470
3A94F	xATANH	454	3B5DA	xENG	458
3A9B7	xEXP	459	3B5FA	xSTD	472
3AA01	xLN	464	3B615	xFS?C	460
3AA73	xLOG	464	3B635	xFC?C	459
3AAE5	xALOG	453	3B655	xBIN	454
3AB2F	xLNP1	464	3B670	xDEC	456
3AB6F	xEXPM	459	3B68B	XHEX	461
3ABAF	xFACT	475	3B6A6	XOCT	465
3ABFD	preFACT		3B6C1	xSTWS	472
3AC3D	xIP	462	3B6FA	xRCWS	468
3AC87	xFP	459	3B715	xRCLF	468
3ACD1	xFLOOR	459	3B749	xSTOF	472
3AD1B	xCEIL	455	3B76C	DOSTOALLF2	176
3AD65	xXPON	476	3B7AD	#BBBB	20
3ADA5	XMAX	464	3B7D2	x>LIST	464
3AE2B	xMIN	465	3B7ED	xR>C	468
3AEB1	xRND	469	3B819	xRE	469
3AF3E	xTRNC	474	3B87E	xIM	462
3AFCB	xMOD	465	3B8D7	xSUB	472
3B02E	xMANT	464	3B8F5	#C55	20
3B06E	xD>R	458	3B904	#C22	20
3B0AE	xR>D	468	3B913	#455	18
3B0EC	x>HMS	461	3B928	#411	17
3B10C	$\mathtt{xHMS}{>}$	461	3B93D	#415	17
3B12C	xHMS+	461	3B952	#451	18
3B14C	xHMS-	461	3B967	#855	19
3B16C	xRNRM	469	3B976	#822	19
3B193	xCNRM	455	3B9D2	xREPL	469
3B1BA	xDET	457	3B9FA	#313	17
3B1E1	xDOT	458	3BA09	#515	18
3B208	xCROSS	456	3BA18	#454	18
3B22F	xRSD	470	3BA2D	#414	17
3B251	x%	476	3BAC1	xLIST>	463
3B2DC	x%T	474	3BADA	INNERCOMP>%	71
3B362	x%CH	455	3BAF5	xC>R	456
3B3E6	xRAND	468	3BB1F	xSIZE	471
3B401	xRDZ	469	3BB94	xPOS	467
3B423	xCOMB	456	3BBBE	x>STR	472

3A5D0 - 3D1C7 591

BBBD9	Addr.	Name	Page	Addr.	Name	Page
38C19	3BBD9	xSTR>	472	3C638	xPIXOFF	466
3BC39	3BBF9	xNUM	465	3C662	xPIX?	466
3BC43	3BC19	xCHR	455	3C68C	xLINE	463
BD4C	3BC39	xTYPE	474	3C6B6	XTLINE	473
3BD65	3BC43	XEQTYPE	199	3C6E0	xBOX	454
3BDB2	3BD4C	#AF	16	3C70A	xBLANK	454
SBDE6   XEQ>   458   3C7D8   XGXOR   461	3BD65	#CF	16	3C72A	xPICT	466
3BE38	3BDB2	xVTYPE	475	3C74A	xGOR	460
3BE9B	3BDE6	xEQ>	458	3C7D8	xGXOR	461
3BEC5	3BE38	xOBJ>	465	3C7E2	#C5C	20
3BEEC	3BE9B	x>ARRY	454	3C800	#C2C	20
3BF77	3BEC5	xARRY>	454	3C81E	#85C	19
3C02E	3BEEC	xRDM	468	3C83C	#82C	19
3C084	3BF77	xCON	456	3C866	xLCD>	463
3COBF	3C02E	xIDN	462	3C881	x>LCD	463
3C10F	3C084	xTRN	474	3C8A1	x>GROB	460
3C11E       #410       17       3C8DF       #5B11       20         3C139       ×PUTI       467       3C8FA       ×TEXT       473         3C16B       #750       19       3C915       ×XRNG       476         3C17A       #710       19       3C935       ×YRNG       476         3C1C7       ×GET       460       3C955       ×FUNCTION       460         3C2D       ×GETI       460       3C967       ×CONIC       456         3C2AC       ×V>       475       3C98B       ×PARAMETRIC       466         3C2D6       ×>V2       475       3C98B       ×PARAMETRIC       466         3C3DA       ×>V3       475       3C98B       ×PARAMETRIC       466         3C3DA       ×>V3       475       3C98D       ×TEUTH       474         3C372       ×PMIN       466       3C9C1       ×HISTOGRAM       461         3C392       ×PMAX       466       3C9D3       ×BAR       454         3C3B2       ×AXES       454       3C9E5       ×SAME       470         3C3DC       ×CENTR       455       3CA07       ×AND       453         3C41A       ×RES<	3C0BF	xPUT	467	3C8C6	xARC	453
3C139         XPUTI         467         3C8FA         XTEXT         473           3C16B         #750         19         3C915         XXRNG         476           3C17A         #710         19         3C935         XYRNG         476           3C1C7         XGET         460         3C955         XFUNCTION         460           3C2DC         XGETI         460         3C955         XFUNCTION         466           3C2AC         XV>         475         3C979         XPOLAR         466           3C2AC         XV2         475         3C98B         XPARAMETRIC         466           3C3OA         X>V3         475         3C99D         XTRUTH         474           3C33E         XINDEP         462         3C9AF         XSCATTER           3C372         XPMIN         466         3C99D         XTRUTH         474           3C392         XPMAX         466         3C99D         XBAR         454           3C3B2         XAXES         454         3C9E5         XSAME         470           3C3DC         XCENTR         455         3CAO7         XAND         453           3C41A         XRES	3C10F	#450	17	3C8D0	#2111	20
3C16B	3C11E	#410	17	3C8DF	#5B11	20
3C17A	3C139	xPUTI	467	3C8FA	xTEXT	473
3C1C7       xGET       460       3C955       xFUNCTION       460         3C22D       xGETI       460       3C967       xCONIC       456         3C2AC       xV>       475       3C979       xPOLAR       466         3C2D6       x>V2       475       3C98B       xPARAMETRIC       466         3C30A       x>V3       475       3C99D       xTRUTH       474         3C33E       xINDEP       462       3C9AF       xSCATTER         3C372       xPMIN       466       3C9C1       xHISTOGRAM       461         3C392       xPMAX       466       3C9D3       xBAR       454         3C3B2       xAXES       454       3C9E5       xSAME       470         3C3DC       xCENTR       455       3CA07       xAND       453         3C41A       xRES       469       3CA8D       xOR       466         3C444       x*H       470       3CB13       xNOT       465         3C464       x*W       470       3CB7A       xXOR       475         3C484       xDRAW       458       3CB7A       xXOR       476         3C4BA       xDRAX       458	3C16B	#750	19	3C915	xXRNG	476
3C22D       xGETI       460       3C967       xCONIC       456         3C2AC       xV>       475       3C979       xPOLAR       466         3C2D6       x>V2       475       3C98B       xPARAMETRIC       466         3C30A       x>V3       475       3C99D       xTRUTH       474         3C33E       xINDEP       462       3C9AF       xSCATTER         3C372       xPMIN       466       3C9C1       xHISTOGRAM       461         3C392       xPMAX       466       3C9C1       xHISTOGRAM       451         3C392       xPMAX       466       3C9C1       xHISTOGRAM       461         3C30C       xCENTR       455       3CA07       xAND       453         3C41A       xRES       469       3CA8D       xOR       466         3C44       x*W       470       3CB13       xNOT       465         3C464       x*W <td>3C17A</td> <td>#710</td> <td>19</td> <td>3C935</td> <td>xYRNG</td> <td>476</td>	3C17A	#710	19	3C935	xYRNG	476
3C2AC       xV>       475       3C979       xPOLAR       466         3C2D6       x>V2       475       3C98B       xPARAMETRIC       466         3C30A       x>V3       475       3C99D       xTRUTH       474         3C33E       xINDEP       462       3C9AF       xSCATTER         3C372       xPMIN       466       3C9C1       xHISTOGRAM       461         3C392       xPMAX       466       3C9D3       xBAR       454         3C3B2       xAXES       454       3C9E5       xSAME       470         3C3DC       xCENTR       455       3CA07       xAND       453         3C41A       xRES       469       3CA8D       xOR       466         3C41A       xRES       469       3CA8D       xOR       466         3C444       x*H       470       3CB13       xNOT       465         3C464       x*W       470       3CB7A       xXOR       475         3C484       xDRAW       458       3CBF6       x==       476         3C49F       xAUTO       454       3CD21       x#?       476         3C4BA       xDRAX       458       3CE	3C1C7	xGET	460	3C955	xFUNCTION	460
3C2D6     x>V2     475     3C98B     xPARAMETRIC     466       3C30A     x>V3     475     3C99D     xTRUTH     474       3C33E     xINDEP     462     3C9AF     xSCATTER       3C372     xPMIN     466     3C9C1     xHISTOGRAM     461       3C392     xPMAX     466     3C9D3     xBAR     454       3C3B2     xAXES     454     3C9E5     xSAME     470       3C3DC     xCENTR     455     3CA07     xAND     453       3C41A     xRES     469     3CA8D     xOR     466       3C444     x*H     470     3CB13     xNOT     465       3C464     x*W     470     3CB7A     xXOR     475       3C484     xDRAW     458     3CBF6     x==     476       3C49F     xAUTO     454     3CD21     x#?     476       3C4BA     xDRAX     458     3CE42     x<	3C22D	xGETI	460	3C967	xCONIC	456
3C30A     x>V3     475     3C99D     xTRUTH     474       3C33E     xINDEP     462     3C9AF     xSCATTER       3C372     xPMIN     466     3C9C1     xHISTOGRAM     461       3C392     xPMAX     466     3C9D3     xBAR     454       3C3B2     xAXES     454     3C9E5     xSAME     470       3C3DC     xCENTR     455     3CA07     xAND     453       3C41A     xRES     469     3CA8D     xOR     466       3C444     x*H     470     3CB13     xNOT     465       3C464     x*W     470     3CB7A     XXOR     475       3C484     xDRAW     458     3CBF6     x==     476       3C49F     xAUTO     454     3CD21     x#?     476       3C4BA     xDRAX     458     3CE42     x<	3C2AC	$\times V >$	475	3C979	xPOLAR	466
3C33E       XINDEP       462       3C9AF       XSCATTER         3C372       XPMIN       466       3C9C1       XHISTOGRAM       461         3C392       XPMAX       466       3C9D3       XBAR       454         3C3B2       XAXES       454       3C9E5       XSAME       470         3C3DC       XCENTR       455       3CA07       XAND       453         3C41A       XRES       469       3CA8D       XOR       466         3C41A       XRES       469       3CA8D       XOR       466         3C41A       XRES       469       3CA8D       XOR       466         3C444       X*H       470       3CB13       XNOT       465         3C464       X*W       470       3CB7A       XXOR       475         3C484       XDRAW       458       3CBF6       X==       476         3C49F       XAUTO       454       3CD21       X#?       476         3C4BA       XDRAX       458       3CE42       X       476         3C4F5       XPDIM       466       3CF80       X<=?	3C2D6	x>V2	475	3C98B	xPARAMETRIC	466
3C372       XPMIN       466       3C9C1       XHISTOGRAM       461         3C392       XPMAX       466       3C9D3       XBAR       454         3C3B2       XAXES       454       3C9E5       XSAME       470         3C3DC       XCENTR       455       3CA07       XAND       453         3C41A       XRES       469       3CA8D       XOR       466         3C444       X*H       470       3CB13       XNOT       465         3C464       X*W       470       3CB7A       XXOR       475         3C484       XDRAW       458       3CBF6       X==       476         3C49F       XAUTO       454       3CD21       X#?       476         3C4BA       XDRAX       458       3CE42       X       476         3C4D5       XSCALE       470       3CEE1       X>       476         3C4F5       XPDIM       466       3CF80       X<=?	3C30A	x>V3	475	3C99D	xTRUTH	474
3C392       xPMAX       466       3C9D3       xBAR       454         3C3B2       xAXES       454       3C9E5       xSAME       470         3C3DC       xCENTR       455       3CA07       xAND       453         3C41A       xRES       469       3CA8D       xOR       466         3C444       x*H       470       3CB13       xNOT       465         3C464       x*W       470       3CB7A       xXOR       475         3C484       xDRAW       458       3CBFA       x==       476         3C49F       xAUTO       454       3CD21       x#?       476         3C4BA       xDRAX       458       3CE42       x<	3C33E	XINDEP	462	3C9AF	XSCATTER	
3C3B2       xAXES       454       3C9E5       xSAME       470         3C3DC       xCENTR       455       3CA07       xAND       453         3C41A       xRES       469       3CA8D       xOR       466         3C444       x*H       470       3CB13       xNOT       465         3C464       x*W       470       3CB7A       xXOR       475         3C484       xDRAW       458       3CBF6       x==       476         3C49F       xAUTO       454       3CD21       x#?       476         3C4BA       xDRAX       458       3CE42       x<	3C372	xPMIN	466	3C9C1	xHISTOGRAM	461
3C3DC       xCENTR       455       3CA07       xAND       453         3C41A       xRES       469       3CA8D       xOR       466         3C444       x*H       470       3CB13       xNOT       465         3C464       x*W       470       3CB7A       xXOR       475         3C484       xDRAW       458       3CB7A       xXOR       476         3C49F       xAUTO       454       3CD21       x#?       476         3C4BA       xDRAX       458       3CE42       x<	3C392	xPMAX	466	3C9D3	xBAR	454
3C41A       XRES       469       3CA8D       XOR       466         3C444       x*H       470       3CB13       XNOT       465         3C464       x*W       470       3CB7A       XXOR       475         3C484       XDRAW       458       3CBF6       X==       476         3C49F       XAUTO       454       3CD21       X#?       476         3C4BA       XDRAX       458       3CE42       X       476         3C4D5       XSCALE       470       3CEE1       X>       476         3C4F5       XPDIM       466       3CF80       X<=?	3C3B2	xAXES	454		xSAME	470
3C444       x*H       470       3CB13       xNOT       465         3C464       x*W       470       3CB7A       xXOR       475         3C484       xDRAW       458       3CBF6       x==       476         3C49F       xAUTO       454       3CD21       x#?       476         3C4BA       xDRAX       458       3CE42       x<	3C3DC	xCENTR	455	3CA07	xAND	453
3C464       x*W       470       3CB7A       xXOR       475         3C484       xDRAW       458       3CBF6       x==       476         3C49F       xAUTO       454       3CD21       x#?       476         3C4BA       xDRAX       458       3CE42       x<	3C41A	xRES	469	3CA8D	xOR	
3C484       xDRAW       458       3CBF6       x==       476         3C49F       xAUTO       454       3CD21       x#?       476         3C4BA       xDRAX       458       3CE42       x       476         3C4D5       xSCALE       470       3CEE1       x>       476         3C4F5       xPDIM       466       3CF80       x<=?	3C444	x*H	470	3CB13	xNOT	
3C49F       xAUTO       454       3CD21       x#?       476         3C4BA       xDRAX       458       3CE42       x       476         3C4D5       xSCALE       470       3CEE1       x>       476         3C4F5       xPDIM       466       3CF80       x<=?	3C464	x*W	470	3CB7A	xXOR	
3C4BA       xDRAX       458       3CE42       x       476         3C4D5       xSCALE       470       3CEE1       x>       476         3C4F5       xPDIM       466       3CF80       x<=?	3C484	xDRAW	458	3CBF6	X==	476
3C4D5       xSCALE       470       3CEE1       x>       476         3C4F5       xPDIM       466       3CF80       x<=?	3C49F	xAUTO	454	3CD21	x#?	476
3C4F5       xPDIM       466       3CF80       x<=?	3C4BA	xDRAX	458	3CE42	x<	476
3C51F       xDEPND       457       3D01F       x>=?       476         3C553       xERASE       458       3D0BC       xOLDPRT         3C56E       xPX>C       467       3D0D7       xPR1       467         3C58E       xC>PX       456       3D0F2       xPRSTC         3C5AE       xGRAPH       460       3D10D       xPRST       467         3C5C9       xLABEL       463       3D128       xCR       456         3C5E4       xPVIEW       467       3D143       xPRVAR       467	3C4D5	xSCALE	470	3CEE1	x>	476
3C553       xERASE       458       3D0BC       xOLDPRT         3C56E       xPX>C       467       3D0D7       xPR1       467         3C58E       xC>PX       456       3D0F2       xPRSTC         3C5AE       xGRAPH       460       3D10D       xPRST       467         3C5C9       xLABEL       463       3D128       xCR       456         3C5E4       xPVIEW       467       3D143       xPRVAR       467	3C4F5	xPDIM		3CF80	x<=?	
3C56E       xPX>C       467       3D0D7       xPR1       467         3C58E       xC>PX       456       3D0F2       xPRSTC         3C5AE       xGRAPH       460       3D10D       xPRST       467         3C5C9       xLABEL       463       3D128       xCR       456         3C5E4       xPVIEW       467       3D143       xPRVAR       467	3C51F	XDEPND	457	3D01F	x>=?	476
3C58E       xC>PX       456       3D0F2       xPRSTC         3C5AE       xGRAPH       460       3D10D       xPRST       467         3C5C9       xLABEL       463       3D128       xCR       456         3C5E4       xPVIEW       467       3D143       xPRVAR       467	3C553	xERASE	458	3D0BC	xOLDPRT	
3C5AE       xGRAPH       460       3D10D       xPRST       467         3C5C9       xLABEL       463       3D128       xCR       456         3C5E4       xPVIEW       467       3D143       xPRVAR       467	3C56E	xPX>C		3D0D7	xPR1	467
3C5C9       xLABEL       463       3D128       xCR       456         3C5E4       xPVIEW       467       3D143       xPRVAR       467	3C58E	xC>PX		3D0F2	xPRSTC	
3C5E4 xPVIEW 467 3D143 xPRVAR 467	3C5AE	xGRAPH		3D10D	xPRST	
	3C5C9	xLABEL		3D128	xCR	
3C60E xPIXON $466 \mid 3D1C7 \mid xDELAY$ $457$	3C5E4	xPVIEW		3D143	xPRVAR	
	3C60E	xPIXON	466	3D1C7	xDELAY	457

Addr.	Name	Page	Addr.	Name	Page
3D1E7	xPRLCD	467	3DE90	xSUMX2	461
3D202	$x\partial$	457	3DEAB	xSUMY2	461
3D258	xDER		3DEC6	xSUMXY	461
3D2B4	CKSYMBTYPE	199	3DEE1	xMAXSIGMA	464
3D393	xRCEQ	468	3DEFC	xMEAN	464
3D3AE	xSTEQ	472	3DF17	xMINSIGMA	464
3D3CE	xROOT	469	3DF32	xSDEV	470
3D434	x∫	462	3DF4D	xTOT	473
3D47E	xINTEGRAL		3DF68	xVAR	475
3D503	xSUM	460	3DF83	xLR	464
3D50D	#A110	20	3DFDD	xPREDV	467
3D51C	#AA10	20	3DFFD	xPREDY	467
3D52B	#A1A0	20	3E01D	xPREDX	467
3D56B	x	460	3E03D	xXCOL	475
3D605	xWHERE		3E05D	xYCOL	476
3D6F6	XQUOTE	468	3E07D	XUTPC	474
3D7AC	xAPPLY	453	3E09D	xUTPN	474
3D81D	xFCNAPPLY		3E0BD	xUTPF	474
3DA3E	x->Q	468	3E0DD	xUTPT	474
3DA63	x->QPI	468	3E0FD	xSIGMACOL	455
3DAD0	xMATCHUP	464	3E127	xSCLSIGMA	470
3DB04	xMATCHDN	475	3E156	xSIGMALINE	460
3DB62	xFORMUNIT		3E171	xBINS	454
3DBCA	xPREDIV	468	3E17B	#111	17
3DBEA	xDUP	458	3E196	xBARPLOT	454
3DC05	xDUP2	458	3E1CA	xHISTPLOT	461
3DC20	xSWAP	473	3E1EF	xSCATRPLOT	470
3DC3B	xDROP	458	3E214	xLINFIT	463
3DC56	xDROP2	458	3E239	xLOGFIT	464
3DC71	xROT	469	3E25E	xEXPFIT	459
3DC8C	XOVER	466	3E283	xPWRFIT	
3DCA7	xDEPTH	457	3E2C1	xBESTFIT	454
3DCC7	xDROPN	458	3E331	xSINV	471
3DCE2	xDUPN	458	3E35B	xSNEG	471
3DCFD	xPICK	466	3E385	xSCONJ	470
3DD18	xROLL	469	3E3AF	xSTO+	472
3DD33	xROLLD	469	3E406	xSTO-	472
3DD4E	xCLEAR	455	3E46C	xSTO/	472
3DD6E	xSTOSIGMA	472	3E4D2	xSTO*	472
3DD8E	xCLSIGMA	455	3E54C	xINCR	462
3DDA9	xRCLSIGMA	468	3E576	xDECR	456
3DDC4	xSIGMA+	460	3E5A0	xCOLCT	455
3DDEE	xSIGMA-	460	3E5E9	xEXPAN	459
3DE09	xNSIGMA	465	3E632	xRULES	100
3DE24	xCORR	456	3E648	xISOL	462
3DE3F	xCOV	456	3E66F	xQUAD	468
3DE5A	xSUMX	461	3E696	xSHOW	471
					473
3DE3A	xSUMY	461	3E6CA	xTAYLR	

3D1E7 - 077002 593

Addr.	Name	Page	Addr.	Name	Page
3E6F1	xRCL	468	3EF07	xSTOKEYS	472
3E739	xSTO	472	3EF3B	xDELKEYS	457
3E743	#9FD	19	3EF79	xRCLKEYS	468
3E759	#8FD	19	3EFB1	x->TAG	473
3E7DA	#C8	16	3EFEF	xDTAG	458
3E7E9	#9F1	19	3F007	xINT	462
3E7FF	#8F1	19	3F033	xANS	453
3E823	xSTO>	167,472	3F053	x;	
3E85C	XDEFINE	456	3F070	xR>I	469
3E87C	xPURGE	467	3F0B7	xI>R	462
3E8C1	$\times$ MEM	464	3F0FC	xNOVAL	465
3E8F0	xORDER	466	3F11C	xCMDAPPLY	
3E91A	xCLUSR	455	3F218	xRPL>	
3E97B	xTMENU	473	3F22E	xUNROT	474
3E9D4	xMENU	464	3F249	xUNPICK	474
3EA01	CST	293	3F264	xNIP	465
3EA2E	xRCLMENU	468	3F27F	xPICK3	466
3EA49	xPVARS	467	3F29A	xDUPDUP	458
3EAA7	xPGDIR	466	3F2B5	xNDUPN	465
3EAC7	xARCHIVE	453	3F2DF	xFAST3D	459
3EAE7	xRESTORE	469	3F481	COERCE2	22
3EAFB	#9F	16	3F495	UNCOERCE2	31
3EB16	xMERGE		4EA22	%TICKSsec	29
3EB2C	xFREE		4EA37	%TICKSmin	29
3EB42	xLIBS	463	4EA4C	%TICKShour	29
3EB64	xATTACH	454	4EA61	%TICKSday	29
3EB84	xDETACH	457	4EA76	%TICKSweek	30
3EC35	xXMIT	475	80058	NEXTIRQ	
3EC55	xSRECV	471	800F5	IRAMBUFF	
3EC75	XOPENIO	466	806FD	EDITLINE	
3EC95	xCLOSEIO	455	8071B	CONTEXT	
3ECB0	xSEND	471	80EDC	DEPTHSAVE	
3ECE4	xKGET	463	80F02	SystemFlags	
3ED22	xRECN	469	80F5A	LASTARGCOUNT	
3ED56	xRECV	469	81006	IOCsave	
3ED76	xFINISH	459	860B8	CurROMBank2	
3ED91	xSERVER	471	860CC	FlashROMTAB2	
3EDAC	xCKSM	455	004002	^RunChooseSimple	244
3EDCC	xBAUD	454	005002	^sysCHOOSE	233
3EDEC	xPARITY	466	007002	^Ck&DoMsgBox	283
3EE0C	xTRANSIO	474	02E002	^DoAlert	282
3EE2C	xKERRM	463	070002	^Choose2	233
3EE47	xBUFLEN	454	072002	^Choose3	233
3EE62	xSTIME	472	073002	^Choose3Save	233
3EE82	xSBRK	470	074002	^Choose3Index	233
3EE9D	xPKT	466	075002	^ChooseDefHandler	233
3EEBD	XINPUT	462	076002	^Choose3CANCL	234
3EEE7	xASN	454	077002	^Choose30K	234

Addr.	Name	Page	Addr.	Name	Page
088002	^SaveHARDBUFF	234	015004	^EQW3ViewRightX	
089002	^RestoreHARDBUFF	234	016004	^EQW3ViewLeft	
09D002	^DoCKeyOK	244	017004	^EQW3ViewRight	
09E002	^DoCKeyCancel	244	018004	^EQW3ViewRightRPL	
09F002	^DoCKeyCheck	244	019004	^EQW3GROB	98
0A0002	^DoCKeyChAll	244	01A004	^EQW3GROBStk	98
0AE002	^DoMKeyOK		01B004	^EQW3CursorOn	
0AF002	^DoKeyCancel		01C004	^EQW3CursorOff	
0B0002	^DoCKeyUnChAll	244	01D004	^EQW3Code	
0B1002	^LEDispItem	245	01E004	^EQW3GROBsys	98
0B2002	^LEDispList	245	01F004	^EQW3GROBmini	98
0B3002	^LEDispPrompt	244	020004	^IfMain	257
0B4002	^DoKeyOK		021004	^IfSetFieldVisible	258
0B5002	^DoKeyEdit		022004	^IfSetSelected	258
0BB002	^GetFieldVals		023004	^IfSetGrob	258
0BC002	^IFEDispField		024004	^IfSetFieldValue	258
0BD002	^DOTVARS{}	169	025004	^IfSetCurrentFieldValue	258
0BE002	^ChangeFocus		026004	^IfGetFieldValue	258
06C003	^laDELROW		027004	^IfGetCurrentFieldValue	258
06D003	^laINSROW		028004	^IfGetFieldMessageHan	258
06E003	^laGPROW		029004	^IfGetFieldType	258
09A003	^WRAP\$	48	02A004	^IfGetFieldObjectsType	258
0A4003	^BRdone		02B004	^IfGetFieldDecompObject	258
0A5003	^BRDispItems		02C004	^IfGetFieldChooseData	258
0A6003	^BRinverse		02D004	^IfGetFieldChooseDecomp	259
0A7003	^BRViewItem		02E004	^IfGetFieldResetValue	259
0AB003	^BRGetItem		02F004	^IfSetFieldResetValue	259
0AC003	^SWAPROWS	344	030004	^IfGetFieldInternalVa	259
001004	^FSTR1	54	031004	^IfDisplayFromData	259
002004	^FSTR2		032004	^IfGetNbFields	259
003004	^FSTR3	54	033004	^IfCheckSetValue	259
004004	^FSTR4	54	034004	^IfCheckFieldtype	259
005004	^FSTR5	54	035004	^IfReset	259
006004	^FSTR6	54	036004	^IfSetField	259
007004	^FSTR7	55	037004	^IfKeyChoose	259
008004	^FSTR8		038004	^IfKeyEdit	260
009004	^FSTR9	55	039004	^IfKeyTypes	260
00A004	^FSTR10		03A004	^IfKeyCalc	260
00B004	^FSTR11		03B004	^IfKeyInvertCheck	260
00C004	^FSTR12		03C004	^IfONKeyPress	260
00D004	^FSTR13	55	03D004	^IfEnterKeyPress	260
00E004	^algparse		03F004	^IfSetHelpString	260
00F004	^algunwrap		040004	^IfSetTitle	260
010004	^EQW3		041004	^IfSetTitle2	
011004	^EQW3Edit	311	042004	^IfMain2	261
012004	^EQW3StartEdit		043004	^IfPutFieldsOnStack	261
013004	^EQW3ViewMargin		044004	^IfSetFieldPos	261
014004	^EQW3ViewLeftX		045004	^IfGetFieldPos	261

088002 - 0DD006 595

Addr.	Name	Page	Addr.	Name	Page
046004	^IfDisplayFromData2		0AE006	^PHFctr	374
047004	^IfSetAllLabelsMessages	261	0AF006	^PHFctr1	374
048004	^IfSetAllHelpStrings	261	0B0006	^PHFctr0	374
049004	^IfCreateTitleGrob		0B1006	^DeCntMulti	375
04A004	^IfInitDepth	261	0B2006	^DoLS	375
04B004	^IfTet		0B3006	^PNFctr	375
04C004	^IfGetPrlgFromTypes	259	0B4006	^PSQFF	375
04D004	^IsUncompressDataString	261	0B5006	^LiftZAdic	375
04E004	^KeyLookup		0B6006	^LFCProd	375
067004	^Filer	188	0В7006	^UFactor	375
068004	^Arbo		0B8006	^UFactor1	375
069004	^RENAME		0B9006	^MonicLf	375
06D004	FILER_MANAGER	188	0BA006	^DemonicLf	375
06E004	FILER_MANAGERTYPE	188	0BB006	^LiftLinear	376
06F004	^FontBrowser		0BC006	^LiftGeneral	376
070004	^BrowseMem.1		0BD006	^UFactorDeg2	376
08E006	^BerlekampP	373	0BE006	^CombineFac	376
08F006	^Berlekamp	373	0BF006	^CombProd	376
090006	^ErrInfRes	403	0C0006	^CombInit	376
091006	^ErrUndefRes	403	0C1006	^CombNext	376
092006	^ErrBadDim	403	0C2006	^RmCombNext	376
093006	^ALG48MSOLV	384	0C3006	^PFactTriv	376
094006	^GMSOLV	384	0C4006	^VarFactor	376
095006	^GBASIS	384	0C5006	^PFactPowCnt	377
096006	^GSOLVE	384	0C6006	^PDivLk	377
097006	^GFACTOR	384	0C7006	^Prime+	332
098006	^GREDUCE		0C8006	^Prime-	332
099006	^REDUCE	384	0C9006	^ZFactor	330
09A006	^FASTREDUCE	384	0CA006	^NFactor	330
09В006	^ONE { } POLY	377	0CB006	^NFactorSpc	330
09C006	^TWO{}POLY	377	0CC006	^DupTypeS?	335
09D006	^THREE{}POLY	377	0CD006	^SFactor	330
09E006	^TWO::POLY	377	0CE006	^SPollard	331
09F006	^::POLY	377	0CF006	^BFactor	331
0A0006	^{}POLY	377	0D0006	^BrentPow	331
0A1006	^>TPOLY	377	0D1006	^ZPrime?	331
0A2006	^>HPOLY	378	0D2006	^ZIsPrime?	331
0A3006	^>TPOLYN	378	0D3006	^SIsPrime?	331
0A4006	^>HPOLYN	378	0D4006	^BIsPrime?	332
0A5006	^MKPOLY	378	0D5006	^BRabin	332
0A6006	^ONE>POLY	378	0D6006	^ZTrialDiv2	332
0A7006	^>POLY	377	0D0000	^ZTrialPrime?	332
0A8006	^ALG48FCTR?	374	0D8006	^ZTrialDiv	332
0A9006	^MFactTriv	374	0D0000	^QMod	372
0AA006	^CheckPNoExt	374	0D3000	^QMODSYMext	012
0AB006	^PPP	374	0DB006	^ModPow	
0AC006	^PFactor	$374 \\ 374$	0DB000	^ZQUOText	
0AC006	^PSqff	$374 \\ 374$	0DC006	^ZMod	329
ONDOO	t DATT	014	סטטעעט	ZIMOU	949

Addr.	Name	Page	Addr.	Name	Page
0DE006	^ZDIVext		10E006	^ZBit?	329
0DF006	^QRoot		10F006	^LOPMext	421
0E0006	^ZSQRT	329	110006	^SWAPRMULT	371
0E1006	^PEvalMod	395	111006	^QMul	371
0E2006	^QAddMod	395	112006	^RMULText	371
0E3006	^QSubMod	395	113006	^RASOP	372
0E4006	^QMulMod	395	114006	^SWAPRSUB	371
0E5006	^QDivMod	395	115006	^QSub	371
0E6006	^QInvMod	395	116006	^RSUBext	371
0E7006	^QGcdMod	396	117006	^SWAPRADD	371
0E8006	^QGcdExMod		118006	^QAdd	371
0E9006	^IsV>V?	399	119006	^RADDext	371
0EA006	^PEvalFast?	380	11A006	^SWAPRDIV	372
0EB006	^PZadic	399	11B006	^RDIVext	372
0EC006	^GCDHEUext	399	11C006	^QDiv	371
0ED006	^H>Z	328	11D006	^R15SIMP	
0EE006	^#>Z	327	11E006	^PPow#	
0EF006	^Z2BIN	22	11F006	^RP#	372
0F0006	^COERCE2Z	22	120006	^MPext	372
0F1006	^Z>S	46	121006	^MP0	423
0F2006	^S>Z	328	122006	^MPEXEC	
0F3006	^S>Z?	328	123006	^RPext	372
0F4006	^Z>ZH	328	124006	^PREPARext	422
0F5006	^R>Z	327	125006	^x+ext	347
0F6006	^Z>R	31	126006	^x-ext	347
0F7006	^DupQIsZero?	334	127006	^x*ext	347
0F8006	^QIsZero?	334	128006	^x=ext	350
0F9006	^DupZIsOne?	334	129006	^x/ext	347
0FA006	^ZIsOne?	334	12A006	^2SYMBINCOMP	72,325
0FB006	^DupZIsNeg?	334	12B006	^x^ext	347
0FC006	^ZIsNeg?	334	12C006	^EXPAND^	347
0FD006	^ListPos	69	12D006	^addtXR00T	362
0FE006	^AppendList	73	12E006	^xssSYMXROOT	
0FF006	^Contains?	137	12F006	^addtMIN	362
100006	^SortList	73	130006	^xssSYMMIN	
101006	^ZTrim	329	131006	^addtMAX	362
102006	^ZAbs	329	132006	^xssSYMMAX	
103006	^PNMax	379	133006	^addt<	363
104006	^LISTMAXext	399	134006	^xssSYM </td <td></td>	
105006	^ZNMax	329	135006	^addt<=	363
106006	^ZNMin	329	136006	^xssSYM<=?	
107006	^ZNLT?	334	137006	^addt>	363
108006	^DISTDIVext	372	138006	^xssSYM>?	
109006	^DupZIsTwo?	334	139006	^addt>=	363
10A006	^DupZIsEven?	334	13A006	^xssSYM>=?	
10B006	^Univar?	380	13B006	^addt==	363
10C006	^SUnivar?	380	13C006	^xssSYM=?	
10D006	^ZBits	329	13D006	^addt!=	363

Addr.	Name	Page	Addr.	Name	Page
13E006	^xssSYM#?		16E006	^DIMLIMITS	64
13F006	^addt%	363	16F006	^CKSAMESIZE	339
140006	^xssSYM%		170006	^DTYPENDO?	339
141006	^addt%CH	363	171006	^DTYPFMAT?	201
142006	^xssSYM%CH		172006	^CKNUMARRY	65
143006	^addt%T	363	173006	^2DMATRIX?	339
144006	^xssSYM%T		174006	^MATRIXDIM	341
145006	^addtMOD	363	175006	^SAMEMATRIX	338
146006	^xssSYMMOD		176006	^SAMEMATSCTYPE	338
147006	^addtTRNC	363	177006	^CKMATRIXELEM	326
148006	^xssSYMTRCXY		178006	^MATRIX2ARRAY	65
149006	^addtRND	363	179006	^MATRIX2LIST	338
14A006	^xssSYMRNDXY		17A006	^LIST2MATRIX	338
14B006	^addtCOMB	363	17B006	^LENMATRIX	343
14C006	^xssSYMCOMB		17C006	^XEQARRY>	65
14D006	^addtPERM	363	17D006	^MATEXPLODE	339
14E006	^xssSYMPERM		17E006	^ARRAY2MATRIX	338
14F006	^addtOR	363	17F006	^XEQ>ARRY	65
150006	^xssSYMOR		180006	^XEQ>ARRAY1	
151006	^addtAND	363	181006	^CKALG	201
152006	^xssSYMAND		182006	^TYPEZ?	200
153006	^addtXOR	363	183006	^DUPTYPEZ?	200
154006	^xssSYMXOR		184006	^CK1Z	198,328
155006	^2LAMBIND	116	185006	^CK2Z	198,328
156006	^3LAMBIND	116	186006	^CK3Z	198,328
157006	^SYMBINCOMP	72, 325, 359	187006	^CK1Cext	201,333
158006	^CKINNERCOMP	72	188006	^C2C%%	37
159006	^DUPCKLEN{}	72	189006	^ZZ2C%%ext	37
15A006	^CKCARCOMP	73	18A006	^Z2%%	31
15B006	^CARCOMPext	372	18B006	^C%>C%%	37
15C006	^RISCH13	387	18C006	^E%%>C%%	37
15D006	^CXRIext	333	18D006	^R2Zext	328
15E006	^RIXCext	37	18E006	^Z2Sext	328
15F006	^IRXCext	37	18F006	^CKFPOLYext	326
160006	^IRXC2		190006	^CK2FPOLY	326
161006	^SWAPNDXF	379	191006	^IDNTLAM?	201
162006	^NDXFext	379	192006	^FLOAT?	201
163006	^SWAPFXND	380	193006	^CKSYMREALCMP	201
164006	^FXNDext	380	194006	^TYPEIDNTLAM?	199
165006	^QXNDext	421	195006	^REAL?	201
166006	^NDXQext	421	196006	^TYPEREALZINT?	201
167006	^TYPEIRRQ?	326,421	197006	^OBJ2REAL	31
168006	^DTYPEIRRQ?	326,421	198006	^METAINT?	78,334
169006	^BESTMATRIXTYPE	65	199006	^METAPOSINT?	78,334
16A006	^{}TO[]	338	19A006	^OBJINT?	334
16B006	^[]TO{}	338	19B006	^OBJPOSINT?	334
16C006	^DUPNULL[]?	339	19C006	^CKINT>0	334
16D006	^MDIMS	64	19D006	^Z>#	22

Addr.	Name	Page	Addr.	Name	Page
19E006	^CLEANIDLAM	326	1CE006	^ckaddt+	360
19F006	^ssSYMDER		1CF006	^ckaddt-	361
1A0006	^SYMDER		1D0006	^VERNUMext	408
1A1006	^DERIVext	387	1D1006	^MENUXYext	409
1A2006	^siSYMDER		1D2006	^SAVECASFLAGS	408
1A3006	^DERIVIDNT	388	1D3006	^SAFEPURGE	167
1A4006	^DERIVIDNT1	388	1D4006	^RESTORECASFLAGS	408
1A5006	^DERIV	388	1D5006	^CASFLAGEVAL	408
1A6006	^METADERIV	388	1D6006	^FLAGEXPAND	411
1A7006	^DO>STRID	54	1D7006	^EXPANDBOTH	
1A8006	^METADEROP		1D8006	^FLAGFACTOR	411
1A9006	^METADER+	388	1D9006	^FLAGLISTEXEC	411
1AA006	^METADER-	388	1DA006	^FLAGSYMBEXEC	411
1AB006	^METADER*	388	1DB006	^FLAGIDNTEXEC	411
1AC006	^METADER/	388	1DC006	^FLAGINTVX	411
1AD006	^METADER^	388	1DD006	^DERVX	412
1AE006	^METADERFCN	388	1DE006	^SOLVEXFLOAT	412
1AF006	^METADERDER	388	1DF006	^SYMLIMIT	412
1B0006	^METADERI4	388	1E0006	^FLAGMATRIXLIMIT	412
1B1006	^METADERI3	388	1E1006	^TAYLOR0	412
1B2006	^METADERIFTE	388	1E2006	^FLAGSERIES	412
1B3006	^DERARG	390	1E3006	^PLOTSTK	
1B4006	^METADEREXP	388	1E4006	^PLOTADD	412
1B5006	^METADERLN	388	1E5006	^FLAGIBP	412
1B6006	^METADERLNP1	388	1E6006	^FLAGPREVAL	412
1B7006	^METADERLOG	389	1E7006	^MATRIXRISCH	412
1B8006	^METADERALOG	389	1E8006	^FLAGRISCH	412
1B9006	^METADERABS	389	1E9006	^FLAGDERIV	412
1BA006	^METADERINV	389	1EA006	^FLAGLAP	412
1BB006	^METADERNEG	389	1EB006	^FLAGILAP	412
1BC006	^METADERSQRT	389	1EC006	^FLAGDESOLVE	412
1BD006	^METADER&NEG	388	1ED006	^FLAGLDSSOLV	412
1BE006	^METADERSQ	389	1EE006	^FLAGLDECSOLV	
1BF006	^METADERSIN	389	1EF006	^FLAGTEXPAND	412
1C0006	^METADERCOS	389	1F0006	^FLAGLIN	412
1C1006	^METADERTAN	389	1F1006	^FLAGTSIMP	413
1C2006	^METADERSINH	389	1F2006	^FLAGLNCOLLECT	413
1C3006	^METADERCOSH	389	1F3006	^FLAGEXPLN	413
1C4006	^METADERTANH	389	1F4006	^FLAGSINCOS	413
1C5006	^METADERASIN	389	1F5006	^FLAGTLIN	413
1C6006	^METADERACOS	389	1F6006	^FLAGTCOLLECT	413
1C7006	^METADERATAN	389	1F7006	^FLAGTRIG	413
1C8006	^METADERASH	389	1F8006	^FLAGTRIGCOS	413
1C9006	^METADERACH	389	1F9006	^FLAGTRIGSIN	413
1CA006	^METADERATH	390	1FA006	^FLAGTRIGTAN	413
1CB006	^pshder*	390	1FB006	^FLAGTAN2SC	413
1CC006	^SQRTINVpshd*	390	1FC006	^FLAGHALFTAN	413
1CD006	^ckaddt*	361	1FD006	^FLAGTAN2SC2	413

19E006 - 25D006 599

Addr.	Name	Page	Addr.	Name	Page
1FE006	^FLAGATAN2S	413	22E006	^CURL	415
1FF006	^FLAGASIN2T	413	22F006	^DIVERGENCE	415
200006	^FLAGASIN2C	413	230006	^LAPLACIAN	415
201006	^FLAGACOS2S	413	231006	^HESSIAN	415
202006	^CK&CONVINT	328	232006	^HERMITE	415
203006	^CK&CONV2INT	328	233006	^TCHEBNOCK	415
204006	^CONVBACK2INT	328	234006	^LEGENDRE	415
205006	^CONVBACKINT	328	235006	^LAGRANGE	415
206006	^STEPIDIV2	413	236006	^FOURIER	415
207006	^FLAGDIV2	413	237006	^SIGNE	400
208006	^FLAGGCD	413	238006	^TABVAR	415
209006	^PEGCD	414	239006	^FLAGDIVPC	416
20A006	^IEGCD		23A006	^FLAGTRUNC	416
20B006	^ABCUV	414	23B006	^FLAGSEVAL	416
20C006	^IABCUV	414	23C006	^XNUM	416
20D006	^FLAGLGCD	414	23D006	^REORDER	416
20E006	^FLAGLCM	414	23E006	^USERLVAR	416
20F006	^FLAGSIMP2	414	23F006	^USERLIDNT	416
210006	^FLAGPARTFRAC	414	240006	^EXLR	356
211006	^FLAGPROPFRAC	414	241006	^ADDTMOD	416
212006	^FLAGPTAYL	414	242006	^MADDTMOD	416
213006	^FLAGHORNER	414	243006	^SUBTMOD	416
214006	^EULER	414	244006	^MSUBTMOD	416
215006	^PA2B2	330	245006	^MULTMOD	416
216006	^FLAGCHINREM	414	246006	^MAT*SCMOD	
217006	^ICHINREM	414	247006	^SC*MATMOD	
218006	^ISPRIME	411	248006	^MAT*MATMOD	
219006	^SOLVE1EQ	414	249006	^DIVMOD	
21A006	^SOLVEMANYEQ	414	24A006	^GCD1MOD	
21B006	^ZEROS1EQ	414	24B006	^INVMOD	
21C006	^ZEROSMANYEQ	415	24C006	^MINVMOD	
21D006	^FCOEF	415	24D006	^FLAGDIV2MOD	
21E006	^FROOTS	415	24E006	^FLAGPOWMOD	
21F006	^FACTORS	415	24F006	^FLAGMPOWMOD	
220006	^DIVIS	415	250006	^EXPAMOD	
221006	^STUDMULT		251006	^FLAGEXPAMOD	
222006	^STUDDIV		252006	^FLAGFACTORMOD	395
223006	^rref	415	253006	^MFACTORMOD	395
224006	^FLAGQXA	345	254006	^RREFMOD	
225006	^FLAGAXQ	346	255006	^KEYEVAL	208
226006	^FLAGGAUSS	346	256006	^LIFCext	395
227006	^FLAGSYLVESTER	346	257006	^EvalNoCKx*	417
228006	^PCAR	346	258006	^EvalNoCKx+	417
229006	^MADNOCK	415	259006	^EvalNoCKx-	417
22A006	^SYSTEM	415	25A006	^EvalNoCKx/	417
22B006	^VANDERMONDE	415	25B006	^EvalNoCKx^	417
22C006	^HILBERTNOCK	415	25C006	^EvalNoCKxCHS	417
22D006	^FLAGJORDAN	345	25D006	^EvalNoCKxINV	417
			1		

Addr.	Name	Page	Addr.	Name	Page
25E006	^EvalNoCKxMOD	417	28E006	^RFACT2ext	383
25F006	^EvalNoCKxPERM	418	28F006	^RFACTSTEP3	383
260006	^EvalNoCKxCOMB	418	290006	^RFACTSTEP5	383
261006	^EvalNoCKxOR	418	291006	^METASOLV	383
262006	^EvalNoCKxAND	418	292006	^METASOLVOUT	
263006	^EvalNoCKxXOR	418	293006	^METASOLV2	383
264006	^EvalNoCKxXROOT	418	294006	^METASOLV4	383
265006	^TABVALext	418	295006	^ADDMULTIPL	384
266006	^TOLISText	418	296006	^FACTOOBJext	384
267006	^FROMLISText	418	297006	^SLVARext	353
268006	^PFEXECext	421	298006	^SIMPLIFY	354
269006	^LOP1ext	421	299006	^SIMPlext	354
26A006	^LOPAext	421	29A006	^SYMEXPAN	354
26B006	^LISTSECOext	421	29B006	^SIMPVAR	354
26C006	^rpnQOBJext	423	29D006	^SIMPIDNT	423
26D006	^CK1TONOext	421	29E006	^RCLALLIDNT	
26E006	^COLCext	355	29F006	^RCL1IDNT	424
26F006	^SYMCOLCT		2A0006	^SIMPSYMBS	354
270006	^COLC1		2A1006	^SYMINTEGRAL	
271006	^COLC2		2A2006	^SIMPUSERFCN	354
272006	^MULMULText	381	2A3006	^EVALUSERFCN	354
273006	^METAMULMULT		2A4006	^SIMP	354
274006	^METAMM2	381	2A5006	^DENOLCMext	358
275006	^COMPLISText	381	2A6006	^METADENOLCM	358
276006	^METACOMPRIM	381	2A7006	^SWPSIMPNDXF	424
277006	^METACOMP0		2A8006	^SIMPNDXFext	424
278006	^METACOMP1	381	2A9006	^SIMPext	354
279006	^ADDLISText	381	2AA006	^SIMPEXTOK	
27A006	^DIVISext	381	2AB006	^MAKEPROFOND	378
27B006	^FACT1ext	381	2AC006	^SLOWSIMP2L	
27C006	^FACTOext	381	2AD006	^SIMPGCDext	354
27D006	^ZFACTO	381	2AE006	^SIMP3ext	354
27E006	^SOLVext	382	2AF006	^SIMP3LISText	
27F006	^FRND	382	2B0006	^SIMP3LSTSLOW	
280006	^BICARREE?	382	2B1006	^LPGCDext	358
281006	^REALBICAR	382	2B2006	^SLOWGCDext	358
282006	^FEVIDENText	377	2B3006	^QGcd	358
283006	^EVIDENText	382	2B4006	^GCDext	
284006	^EVIDSOLV	382	2B5006	^CGCDext	333
285006	^DEG2ext	382	2B6006	^CMODext	424
286006	^METADEG2	382	2B7006	^ZGCDext	329
287006	^METADEG1	382	2B8006	^ZGcd	329
288006	^DEG1	382	2B9006	^TSIMP2ext	354
289006	^FDEG2ext	382	2BA006	^TSIMPext	354
28A006	^PIext	73	2BB006	^TSIMP3ext	355
28B006	^RACTOFACext	383	2BC006	^LASTCOMP	68
28C006	^FACTORACext	383	2BD006	^SQFF2ext	424
28D006	^RFACText	383	2BE006	^PPZ	424

25E006 - 31E006 601

Addr.	Name	Page	Addr.	Name	Page
2BF006	^PZHSTR	424	2EF006	^ipi	420
2C0006	^HORNER1ext	424	2F0006	^metaipi	420
2C1006	^PEval	424	2F1006	^meta-pi	420
2C2006	^RISCHext		2F2006	^metapi/2	420
2C3006	^risch/		2F3006	^metapi/4	420
2C4006	^rischABS		2F4006	^meta-pi/2	420
2C5006	^IBP	390	2F5006	^meta-pi/4	420
2C6006	^SQRT_IN?	424	2F6006	^pifois2	420
2C7006	^IS_SQRT?	424	2F7006	^deuxipi	420
2C8006	^XROOT_IN?		2F8006	^metapi*2	420
2C9006	^IS_XROOT?	424	2F9006	^base_ln	420
2CA006	^STOPRIMIT	424	2FA006	^meta_e	420
2CB006	^CONTAINS_LN?	424	2FB006	^NEXTPext	72
2CC006	^ISNT_IDNT?		2FC006	^INSERT{}N	72
2CD006	^RISCHPF		2FD006	^COMPRIMext	73
2CE006	^RISCHRAT		2FE006	^TCOLLECT	355
2CF006	^rischlogpart		2FF006	^SIGMAEXPext	355
2D0006	^PREVALext	390	300006	^LINEXPext	355
2D1006	^WARNSING	390	301006	^SIGMAEXP2ext	355
2D2006	^INText	390	302006	^TCHEBext	378
2D3006	^INT3	390	303006	^SINEXPA	355
2D4006	^FOURIERext	424	304006	^METASINEXPA	368
2D5006	^3DUP	106	305006	^SINEXPA+	368
2D6006	^#3+ROLL	108	306006	^SINEXPA-	368
2D7006	^2DROPTRUE	136	307006	^SINEXPA*	369
2D8006	^IRRQ#ULTIMATE	422	308006	^SINEXPA*1	369
2D9006	^LESSCOMPLEX?	424	309006	^COSEXPA	355
2DA006	^LISTIRRQ	422	30A006	^METACOSEXPA	369
2DB006	^LIST1i-1-i		30B006	^COSEXPA+	369
2DC006	^LIST10-10		30C006	^COSEXPA-	369
2DD006	^TABLECOSext	425	30D006	^COSEXPA*	369
2DE006	^TABLETANext	425	30E006	^COSEXPA*1	369
2DF006	^DROPZ1	327	30F006	^EXPEXPA	355
2E0006	^DROPZ0	327	310006	^METAEXPEXPA	369
2E1006	^TESTINFINI	419	311006	^EXPEXPA+	369
2E2006	^INFINIext	419	312006	^EXPEXPA-	369
2E3006	^MINUSINFext	419	313006	^EXPEXPA*	369
2E4006	^PLUSINFext	419	314006	^EXPEXPANEG	369
2E5006	^?ext	419	315006	^EXPEXPA*1	369
2E6006	^POSINFext	419	316006	^LNEXPA	355
2E7006	^POSUNDEFext	420	317006	^METALNEXPA	369
2E8006	^pisur2	420	318006	^LNEXPA*	369
2E9006	^pisur-2	420	319006	^LNEXPA/	369
2EA006	^pi	420	31A006	^LNEXPA^	369
2EB006	^metapi	420	31B006	^LINEXPA	355
2EC006	^'xPI	420	31C006	^MTRIG2SYMB	355
2ED006	^metai	420	31D006	^LNCOLCext	355
2EE006	^'xi	420	31E006	^METATANEXPA	369

Addr.	Name	Page	Addr.	Name	Page
31F006	^TEXPAext	356	34F006	^MATREFRREF	341
320006	^MAT+	339	350006	^INXREDext	341
321006	^MADD	339	351006	^METAMATRED	341
322006	^MAT-	339	352006	^METAPIVOT	341
323006	^MSUB	339	353006	^PIVOTNORM	
324006	^VADD	339	354006	^PIVOTFLOAT	341
325006	^VSUB	339	355006	^SYSText	342
326006	^MAT*	339	356006	^STOSYSText	342
327006	^MMMULT	339	357006	^MAKESYSText	342
328006	^MVMULT	339	358006	^VARGENext	
329006	^SCL*MAT	339	359006	^NULLVECTOR?	339
32A006	^MAT*SCL	339	35A006	^FINDELN	342
32B006	^VPMULT	339	35B006	^PULLEL[S]	342
32C006	^MAT^	340	35C006	^BANGARRY	343
32D006	^MATCROSS	340	35D006	^PUT[]	343
32E006	^MATDOT	340	35E006	^ARSIZE	64
32F006	^RNDARRY	340	35F006	^MATRIX>DIAG	343
330006	^TRCARRY	340	360006	^MATRIXDIAG>	343
331006	^Yext	65	361006	^la+ELEMsym	343
332006	^MAT/SCL	340	362006	^INSERTROW[]	343
333006	^MAT/	340	363006	^insertrow[]	343
334006	^MATCHS	340	364006	^INSERTCOL[]	344
335006	^MATSQUARE	340	365006	^INSERT[]ROW[]	344
336006	^MATCONJ	340	366006	^INSERT[]COL[]	344
337006	^MATRE	340	367006	^MATRIXRCI	341
338006	^MATIM	340	368006	^MATRIXRCIJ	341
339006	^MATTRACE	340	369006	^MATRIXCSWAP	344
33A006	^MATTRN	340	36A006	^MATRIXRSWAP	344
33B006	^MATTRAN	65	36B006	^MATRIX-ROW	344
33C006	^mattran	340	36C006	^METAMAT-ROW	344
33D006	^mattrn	340	36D006	^MATRIX-COL	344
33E006	^MATSUB	343	36E006	^METAMATCSWAP	344
33F006	^submeta	78	36F006	^METAMATRSWAP	344
340006	^MATREPL	343	370006	^STOMAText	344
341006	^MATREDIM	337	371006	^MATIDN	337
342006	^VRRDM	338	372006	^MATCON	337
343006	^VRRDMmeta	338	373006	^MAKEARRY	337
344006	^MATRANM	337	374006	^OBJDIMS2MAT	337
345006	^DIMRANM	337	375006	^LCPROG2M	337
346006	^MATDET	340	376006	^MAKE2DMATRIX	337
347006	^MATRDET	340	377006	^make2dmatrix	337
348006	^MATFNORM	340	378006	^ADDMATOBJext	345
349006	^MATRNORM	340	379006	^VUNARYOP	345
34A006	^MATCNORM	340	37A006	^VBINARYOP	345
34B006	^MATRREF	341	37B006	^PEVAL	345
34C006	^MATREF	341	37C006	^MATEGVL	345
34D006	^MATRANK	341	37D006	^ROOTM2ROOT	385
34E006	^MATINV	340	37E006	^MADJ	345

31F006 - 3DE006 603

Addr.	Name	Page	Addr.	Name	Page
37F006	^MATEGV	345	3AF006	^MetaMul	361
380006	^JORDAN	345	3B0006	^xssSYM*	
381006	^QXA	345	3B1006	^MetaDiv	361
382006	^AXQ	345	3B2006	^xssSYM/	
383006	^GAUSS	346	3B3006	^NDROPZ0	327
384006	^SYLVESTER	346	3B4006	^NDROPZ1	327
385006	^metasplit	78	3B5006	^MetaPow	362
386006	^m-1&m+1	359	3B6006	^xssSYM^	
387006	^meta1/meta	359	3B7006	^MetaNeg	362
388006	^1&meta	359	3B8006	^xSYMCHS	
389006	^meta/2	359	3B9006	^metaneg	362
38A006	^addt2	359	3BA006	^metackneg	362
38B006	^addt/	359	3BB006	^metasimp	368
38C006	^meta2*	359	3BC006	^metapi?	370
38D006	^metal-sq	360	3BD006	^metaCOMPARE	370
38E006	^metasq+1	360	3BE006	^STRICTmetaCOMPARE	370
38F006	^metasq-1	360	3BF006	^EQUALPOSMETA	78
390006	^meta-1	360	3C0006	^EQUALPOS2META	78
391006	^NDROPZERO	76	3C1006	^vgerxssSYMSUM	394
392006	^2DROPZ0	327	3C2006	^DISTRIB/	368
393006	^metaadd	360	3C3006	^metareal?	370
394006	^metasub	360	3C4006	^ModExpa	372
395006	^metamult	361	3C5006	^ModAdd	372
396006	^metadiv	361	3C6006	^ModSub	373
397006	^meta^	361	3C7006	^ModMul	373
398006	^addt^	360	3C8006	^ModDiv	373
399006	^metapow	361	3C9006	^ModDiv2	373
39A006	^metafraction?	370	3CA006	^ModInv	373
39B006	^metaxroot	362	3CB006	^ModGcd	373
39C006	^top&addt*	360	3CC006	^ModLGCD	
39D006	^top&addt/	360	3CD006	^ModLOPD	
39E006	^addti	360	3CE006	^MODULOMODext	
39F006	^metaEQUAL?	78	3CF006	^MODULOMAText	
3A0006	^2metaundef#	367	3D0006	^Mod	329
3A1006	^1metaundef#	367	3D1006	^ModFctr	
3A2006	^metaundef	367	3D2006	^PARTFRAC	391
3A3006	^2metainf#	368	3D3006	^INPARTFRAC	391
3A4006	^1metainf#	367	3D4006	^PARTFRACRAT	
3A5006	^metainftype	368	3D5006	^PFext	
3A6006	^unsignedinf	368	3D6006	^IEGCDext	330
3A7006	^plusinf	368	3D7006	^REGCDext	380
3A8006	^NDROPplusinf	368	3D8006	^EGCDext	380
3A9006	^minusinf	368	3D9006	^INEGCD	330
3AA006	^NDROPminusinf	368	3DA006	^EGCDSWAP	
3AB006	^MetaAdd	360	3DB006	^EGCDNEWG	
3AC006	^xssSYM+		3DC006	^PDer	387
3AD006	^MetaSub	361	3DD006	^INTEGRext	391
3AE006	^xssSYM-		3DE006	^LRDMext	378

Addr.	Name	Page	Addr.	Name	Page
3DF006	^RRDMext	378	40F006	^tan2tan/2	366
3E0006	^DEGREext	378	410006	^addtTAN/2	366
3E1006	^FHORNER	378	411006	^TRIGTAN	356
3E2006	^HORNext	378	412006	^COS2TAN	351
3E3006	^HORN1		413006	^cos2tan	366
3E4006	^MHORNext	378	414006	^SIN2TAN	351
3E5006	^PTAYLext	372	415006	^sin2tan	366
3E6006	^LAGRANGEext	379	416006	^TRIGext	356
3E7006	^PSEUDOPREP	422	417006	^HYP2EXPext	356
3E8006	^PSEUDODIV	357	418006	^EXPLNext	356
3E9006	^IDIV2		419006	^SERIESEXPLN	356
3EA006	^BESTDIV2	357	41A006	^LNP12LN	351
3EB006	^CDIV2ext		41B006	^LOG2LN	351
3EC006	^QUOText	357	41C006	^ALOG2EXP	351
3ED006	^NEWDIVext	357	41D006	^EXPM2EXP	351
3EE006	^QDivRem	372	41E006	^SQRT2LNEXP	351
3EF006	^DIV2LISText	372	41F006	^sqrt2lnexp	351
3F0006	^DIVOBJext	420	420006	^TAN2EXP	351
3F1006	^DIVMETAOBJ	88, 361	421006	^tan2exp	366
3F2006	^LOPDext	420	422006	^ASIN2LN	351
3F3006	^QUOTOBJext	357	423006	^asin2ln	366
3F4006	^DIVISIBLE?	357	424006	^ACOS2LN	351
3F5006	^QDiv?	357	425006	^acos2ln	366
3F6006	^FastDiv?	357	426006	^TAN2SCext	356
3F7006	^POTENCEext	358	427006	^TAN2SC	351
3F8006	^PDIV2ext	372	428006	^sin/cos	366
3F9006	^PSetSign	372	429006	^SIN2TCext	356
3FA006	^PLCZ		42A006	^SIN2TC	351
3FB006	^HSECO2RCext	422	42B006	^cos*tan	366
3FC006	^SECO2CMPext	422	42C006	^COS2ext	351
3FD006	^SECO2CMPPOL		42D006	^sqrt1-sin^2	366
3FE006	^SECO2CMPCART		42E006	^SIN2ext	351
3FF006	^VALOBJext	423	42F006	^sqrt1-cos^2	366
400006	^R2SYM	325	430006	^ATAN2Sext	356
401006	^VAL2ext	423	431006	^ATAN2ASIN	352
402006	^INVAL2	423	432006	^atan2asin	366
403006	^METAVAL2	423	433006	^ASIN2Text	356
404006	^VAL1	423	434006	^ASIN2ATAN	352
405006	^VAL1M	423	435006	^asin2atan	366
406006	^addt0meta	77	436006	^ASIN2Cext	357
407006	^HALFTAN	356	437006	^ASIN2ACOS	352
408006	^COS2TAN/2	351	438006	^pi/2-acos	366
409006	^cos2tan/2	365	439006	^pi/2-meta	366
40A006	$1-x^2/1+x^2$	365	43A006	^ACOS2Sext	357
40B006	^SIN2TAN/2	351	43B006	^pi/2-asin	366
40C006	^sin2tan/2	365	43C006	^ACOS2ASIN	352
40D006	$2x/1+x^2$	365	43D006	^ATAN2LNext	352
40E006	^TAN2TAN/2	351	43E006	^atan2ln	366

3DF006 - 49F006 605

Addr.	Name	Page	Addr.	Name	Page
43F006	^TAN2SC2ext	357	470006	^SYMPAPRX	
440006	^TAN2SC2	352	471006	^TRUNCDL	386
441006	^2*1-cos/sin	366	472006	^LIMSERIES!	386
442006	^TAN2CS2	352	473006	^LIMITX!	
443006	^2*sin/1+cos	367	474006	^LIMITNOVX!	
444006	^SIN2EXPext	352	475006	^LIMERRO!	
445006	^sin2exp	367	476006	^LIMERR1!	
446006	^COS2EXPext	352	477006	^LIMIT!	386
447006	^cos2exp	367	478006	^LIMSTEP1!	386
448006	^SINH2EXPext	352	479006	^LIMSTEP2!	
449006	^sinh2exp	367	47A006	^LIMSTEP3!	
44A006	^COSH2EXPext	352	47B006	^LIMSTEP4!	
44B006	^cosh2exp	367	47C006	^LIMLIM!	386
44C006	^TANH2EXPext	352	47D006	^n{}N	
44D006	^tanh2exp	367	47E006	^LIMLIM1!	
44E006	^ASINH2LNext	352	47F006	^LIMCMPL!	386
44F006	^asinh2ln	367	480006	^LIMEQUFR!	386
450006	^ACOSH2LNext	352	481006	^LIMEQU!	386
451006	^acosh2ln	367	482006	^LIMEQUO!	
452006	^ATANH2LNext	352	483006	^LIM+-!	387
453006	^atanh2ln	367	484006	^LIMERR10!	
454006	^XROOT2ext	352	485006	^LIMNEG!	
455006	^xroot2expln	367	486006	^LIMRAC!	
456006	^LN2ext	357	487006	^LIMINV!	
458006	^exp2sincos	367	488006	^LIM/!	
459006	^metai*	360	489006	^LIMPOW!	
45A006	^LN2ATAN	352	48A006	^LIMSQ!	
45B006	^VAR=LIST	353	48B006	^LIM*!	
45C006	^IDNTEXEC	423	48C006	^LIMDIVPC!	387
45D006	^SYMISOL		48D006	^DIVPC!	
45E006	^SYMQFORM		48E006	^LIMPROFEND!	387
45F006	^LISTEXEC	421	48F006	^LIMPROF!	
460006	^LISTEXEC1	421	490006	^LIM%#!	387
461006	^SECOEXEC	421	491006	^LIMPROF0!	
462006	^EQUATION?	86	492006	^LIMPROF1!	
463006	^USERFCN?	86	493006	^LIMPROF2!	
464006	^SYMBEXEC	353	494006	^LIMINVLN!	
465006	^MEVALext	353	495006	^LIMLN!	
466006	^CASNUMEVAL	353	496006	^LIMEXP!	
467006	^CASCOMPEVAL	353	497006	^LIMSINCOS!	
468006	^REPLACE2BY1	353	498006	^LIMATAN!	
469006	^NR_REPLACE	353	499006	^LIMASIN!	
46A006	^SYMBWHERE	333	49A006	^LIMSQRT!	
46B006	^CASCRUNCH	353	49B006	^LIMFLOOR!	
46C006	^APPROXCOMPEVAL	353	49C006	^LIMABS!	
46D006	^LIMIText	333	49D006	^LPROF!	
46E006	^REWRITEIFINF		49E006	^LIM#VARX!	387
46F006	^SYMTAYLOR	386	49F006	^LIMBETA!	55.
101 000	211111111010	500	171000		

Addr.	Name	Page	Addr.	Name	Page
4A0006	^LIMALPHA!		4D0006	^MZSQFF1	396
4A1006	^HORNEXP!	387	4D1006	^MSECOSQFF	425
4A2006	^HORNCOS!		4D2006	^MLISTSQFF	397
4A3006	^HORNSIN!		4D3006	^METASQFFext	397
4A4006	^LIMSCO!		4D4006	^SECOSQFFext	333,422
4A5006	^LIMSC1!		4D5006	^CSQFFext	333
4A6006	^HORNATAN!		4D6006	^SUMSQRext	333
4A7006	^LIMATAS!		4D7006	^VXXLext	325
4A8006	^HORNASIN!		4D8006	^METALISTVXXL	325
4A9006	^HORNASIN1!		4D9006	^VXXLFext	325
4AA006	^HORNLN!		4DA006	^VXXL1ext	326
4AB006	^LNOBJ!		4DB006	^VXXL0	326
4AC006	^NEWLIMHORN		4DC006	^VXXL2NR	326
4AD006	^LIMHORN!		4DD006	^VXXL2	326
4AE006	^LRDM!		4DE006	^LIDNText	397
4AF006	^LIMDL!		4DF006	^LVARXNXext	397
4B0006	^LIMDLINF!		4E0006	^ISPOLYNOMIAL?	397
4B1006	^LIMINFSIGN!		4E1006	^2POLYNOMIAL?	397
4B2006	^LIMMAX!		4E2006	^VXINDEP?	397
4B3006	^LIMCOMP!		4E3006	^LVARXNX2ext	
4B4006	^VARCOMP2!		4E4006	^RLVARext	397
4B5006	^LIMSORT!		4E5006	^LLVARDext	397
4B6006	^VARCOMP!	387	4E6006	^VXLVARext	397
4B7006	^VARCOMPLN!		4E7006	^LVARext	397
4B8006	^VARCOMP3!		4E8006	^VX>LVARext	398
4B9006	^VARCOMP31!		4E9006	^VX>	398
4BA006	^VARCOMP32!	387	4EA006	^VX!	398
4BB006	^VARCOMP33!		4EB006	^prepvarlist	73
4BC006	^LIMERR6!		4EC006	^LIDNTLVAR	398
4BD006	^LIMVALOBJ!	387	4ED006	^LISTOPRAC	398
4BE006	^LIMVAL!	387	4EE006	^LISTOPext	398
4BF006	^EQUIV!	387	4EF006	^LISTOPSQRT	398
4C0006	^LVARXNX2!	387	4F0006	^LVARDext	398
4C1006	^SIMP1!		4F1006	^>VARLIST	
4C2006	^FindCurVar	387	4F2006	^DEPTHext	398
4C3006	^LIMVAR!	387	4F3006	^DEPTHOBJext	398
4C4006	^VAR%		4F4006	^TRIMext	378
4C5006	^ISOL1	396	4F5006	^PTrim	378
4C6006	^ISOLALL	396	4F6006	^TRIMOBJext	398
4C7006	^ISOL2ext	396	4F7006	^NEWTRIMext	399
4C8006	^BEZOUTMSOLV	396	4F8006	^>POLYTRIM	399
4C9006	^ROOT{}N	396	4F9006	^ELMGext	399
4CA006	^MHORNER	396	4FA006	^SWAPRNEG	348
4CB006	^MHORNER1	396	4FB006	^QNeg	347
4CC006	^SQFFext	396	4FC006	^RNEGext	347
4CD006	^MSQFF	396	4FD006	^SWAPRRE	348
4CE006	^%1TWO	396	4FE006	^RREext	348
4CF006	^MZSQFF	396	4FF006	^SWAPRIM	348

Addr.	Name	Page	Addr.	Name	Page
500006	^RIMext	348	530006	^xSINHext	350
501006	^xREext	348	531006	^xTANHext	350
502006	^xSYMRE	362	532006	^xASINext	349
503006	^xIMext	348	533006	^xACOSext	349
504006	^xSYMIM	362	534006	^xATANext	349
505006	^RCONJext	348	535006	^addtCOS	363
506006	^addtCONJ	363	536006	^xSYMCOS	349
507006	^xSYMCONJ		537006	^addtSIN	364
508006	^QCONJext	422	538006	^xSYMSIN	349
509006	^QABSext	422	539006	^addtTAN	364
50A006	^RABSext	348	53A006	^xSYMTAN	349
50B006	^ZABS	329	53B006	^addtSINACOS	364
50C006	^CZABS	37	53C006	^addtASIN	364
50D006	^xABSext	348	53D006	^xSYMASIN	349
50E006	^addtABS	362	53E006	^addtACOS	364
50F006	^xSYMABS		53F006	^xSYMACOS	349
510006	^addtABSEXACT	362	540006	^addtATAN	364
511006	^addtSIGN	362	541006	^xSYMATAN	349
512006	^xSYMSIGN		542006	^addtSINH	364
513006	^addtARG	362	543006	^xSYMSINH	350
514006	^xSYMARG		544006	^addtCOSH	364
515006	^ARG2	38	545006	^xSYMCOSH	349
516006	^INTERNALARG2		546006	^addtTANH	364
517006	^QUADRANT	38	547006	^xSYMTANH	350
518006	^CNORMext	333	548006	^xATANHext	350
519006	^CXIRext		549006	^addtATANH	364
51A006	^QNORMext	422	54A006	^xSYMATANH	350
51B006	^SXSQRext	348	54B006	^xASINHext	350
51C006	^XSQRext	348	54C006	^addtASINH	364
51D006	^CK%%SQRT	34	54D006	^xSYMASINH	350
51E006	^C%%SQRT	38	54E006	^xACOSHext	349
51F006	^ZINTSQRT		54F006	^addtACOSH	364
520006	^SHALT		550006	^xSYMACOSH	350
521006	^CKLN	348	551006	^addtSQRT	364
522006	^xLNext	349	552006	^xSYMSQRT	348
523006	^addtLN	363	553006	^xSQext	348
524006	^xSYMLN		554006	^addtSQ	364
525006	^EXPANDLN	349	555006	^xSYMSQ	348
526006	^CMPLXLN	349	556006	^addtINV	364
527006	^LNATANext	349	557006	^xSYMINV	348
528006	^REALLN	349	558006	^addtEXP	364
529006	^xEXPext	349	559006	^xSYMEXP	364
52A006	^xINVext	348	55A006	^addtD->R	364
52B006	^xvext	348	55B006	^xSYMD>R	
52C006	^xCOSext	349	55C006	^addtR->D	365
52D006	^xSINext	349	55D006	^xSYMR>D	
52E006	^xTANext	349	55E006	^addtFLOOR	365
52F006	^xCOSHext	349	55F006	^xSYMFLOOR	350
			1		

Addr.	Name	Page	Addr.	Name	Page
560006	^addtCEIL	365	001007	^ListToArry	65
561006	^xSYMCEIL	350	002007	^ArryToMatrix	65
562006	^addtIP	365	003007	^ArryToList	338
563006	^xSYMIP	350	004007	^DIMS	
564006	^addtFP	365	005007	^RunDoOldMatrix	
565006	^xSYMFP	350	006007	^RunDoNewMatrix	
566006	^addtXPON	365	007007	^DoNewMatrixReal	
567006	^xSYMXPON	350	008007	^DoNewMatrixCplx	
568006	^addtMANT	365	009007	^DoOldMatrixReal	
569006	^xSYMMANT	350	00A007	^DoOldMatrixCplx	
56A006	^addtLNP1	365	00B007	^DoNewMatrixRealOrCplx	
56B006	^xSYMLNP1	350	00C007	^DEB.MATRIX	
56C006	^addtLOG	365	00D007	^DEB.MATRIXTYPE	
56D006	^xSYMLOG	350	073007	^QpiZ	418
56E006	^addtALOG	365	074007	^QPI	418
56F006	^xSYMALOG	350	075007	^QpiSym	419
570006	^addtEXPM	365	076007	^QpiArry	419
571006	^xSYMEXPM1	350	077007	^QpiList	419
572006	^factorial	350	078007	^Qpi	419
573006	^facts	350	079007	^Qpi%	419
574006	^addtFACT	365	07A007	^GetRoot	419
575006	^xSYMFACT	350	07В007	^Approx	419
576006	^factzint	330	07C007	^#FACT	330
577006	^addtNOT	365	07D007	^CHECKSING	402
578006	^xSYMNOT	350	07E007	^DESOLVE	391
579006	^Verbose1	417	07F007	^ODE_INT	390
57A006	^Verbose2	417	080007	^LINSOLV	342
57B006	^Verbose3	417	081007	^LDECSOLV	391
57C006	^VerboseN	417	082007	^LDEGENE	391
57D006	^GETERABLEMSG	403	083007	^LDEPART	391
57E006	^ERABLEERROR	403	084007	^LDSSOLVext	391
57F006	^CANTFACTOR	403	085007	^ODETYPESTO	391
580006	^TRANSCERROR	403	086007	^ODE_SEPAR	392
581006	^NONUNARYERR	403	087007	^LAPext	392
582006	^INTERNALERR	403	088007	^ILAPext	392
583006	^INVALIDOP	403	089007	^ILAPRAText	
584006	^ISOLERR	404	08A007	^ILAPDELTA	
585006	^NONINTERR	404	08B007	^ILAPEXP	392
586006	^INTVARERR	404	08C007	^ILAPEXPSC	
587006	^Z>#ERR	404	08D007	^MENUext	409
588006	^Z<0ERR	404	08E007	^WRITEMENU	293
589006	^VXINDEPERR	404	08F007	^CFGDISPLAY	405
58A006	^NONPOLYSYST	404	090007	^NEWVX	405
58B006	^COMPLEXERR	404	091007	^NEWMODULO	405
58C006	^VALMUSTBE0	404	092007	^SWITCHON	405
58D006	^SWITCHNOTALLOWED	404	093007	^SWITCHOFF	405
58E006	^ERR\$EVALext	404	094007	^FLAGNAME	405
58F006	^Sys1IT	404	095007	^COMPLEXON	406

560006 - 0F5007 609

Addr.	Name	Page	Addr.	Name	Page
096007	^COMPLEXOFF	406	0C6007	^STOMODULO	408
097007	^EXACTON	406	0C7007	^RCLEPS	408
098007	^EXACTOFF	406	0C8007	^ISIDREAL?	408
099007	^COMPLEXMODE	406	0C9007	^ADDTOREAL	408
09A007	^SETCOMPLEX	406	0CA007	^RESETCASCFG	408
09B007	^COMPLEX?	406	0CB007	^FRACPARITY	425
09C007	^REALMODE	406	0CC007	^POLYPARITY	380
09D007	^CLRCOMPLEX	406	0CD007	^PARITYTEST	
09E007	^EXACTMODE	406	0CE007	^COSTEST	
09F007	^SETEXACT	406	0CF007	^SHRINKEVEN	379
0A0007	^NUMMODE	406	0D0007	^SINTEST	
0A1007	^CLREXACT	406	0D1007	^SHRINK2SYM	379
0A2007	^EXACT?	406	0D2007	^SHRINKSYM	379
0A3007	^STEPBYSTEP	407	0D3007	^SHRINK2ASYM	379
0A4007	^NOSTEPBYSTEP	407	0D4007	^SHRINKASYM	379
0A5007	^VERBOSEMODE	407	0D5007	FR2ND%	425
0A6007	^SILENTMODE	407	0D6007	^POLYSYM	380
0A7007	^RECURMODE	407	0D7007	^POLYASYM	380
0A8007	^NONRECMODE	407	0D8007	^P2P#	374
0A9007	^PLUSAT0	407	0D9007	^NDEvalN/D	394
0AA007	^SETPLUSAT0	407	0DA007	^PEvalN/D	394
0AB007	^PLUSATINFTY	407	0DB007	^POSITIFext	401
0AC007	^CLRPLUSAT0	407	0DC007	^SIGNE1ext	400
0AD007	^SPARSEDATA	407	0DD007	^SIGNEext	
0AE007	^FULLDATA	407	0DE007	^SIGNUNDEF	400
0AF007	^RIGORMODE	407	0DF007	^SIGNPLUS	400
0B0007	^SLOPPYMODE	407	0E0007	^SIGNMOINS	400
0B1007	^SLOPPY?	408	0E1007	^SIGNELN	401
0B2007	^MENUCHOOSE?	409	0E2007	^SIGNEEXP	401
0B3007	^MENUCHOOSE	409	0E3007	^SIGNESIN	401
0B4007	^MENUGENE1	409	0E4007	^SIGNECOS	401
0B5007	^MENUBASE1	409	0E5007	^SIGNETAN	401
0B6007	^MENUCMPLX1	409	0E6007	^SIGNEATAN	401
0B7007	^MENUTRIG1	409	0E7007	^SIGNESQRT	401
0B8007	^MENUMAT1	409	0E8007	^SUBSIGNE	401
0B9007	^MENUARIT1	409	0E9007	^SIGNERIGHT	401
0BA007	^MENUSOLVE1	410	0EA007	^SIGNELEFT	401
0BB007	^MENUEXPLN1	410	0EB007	^>SIGNE	401
0BC007	^MENUDIFF1	410	0EC007	^SIGNE>	401
0BD007	^PROMPTSTO1	167	0ED007	^SIGNMULText	401
0BE007	^XGROBext	98	0EE007	^ZSIGNECK	401
0BF007	^GROBADDext	92	0EF007	^SIGNEERROR	404
0C0007	^DISPLAYext	98	0F0007	^ZSIGNE	401
0C1007	^SCROLLext	281	0F1007	^zsigne	402
0C2007	^RCLMODULO	408	0F2007	^PASCAL_NEXTLINE	385
0C3007	^RCLPERIOD	408	0F3007	^DELTAPSOLVE	385
0C4007	^RCLVX	408	0F4007	^SOLVEMETASYST	342
0C5007	^STOVX	408	0F5007	^REDUCEMETASYST	342
			1		

Addr.	Name	Page	Addr.	Name	Page
0F6007	^REDUCEMETAPSYST	342	0080AB	xWIREFRAME	475
0F7007	^SOLVECRAMER	342	0090AB	xPARSURFACE	466
0F8007	^QUOTExSIGMA		00A0AB	xGRIDMAP	460
0F9007	^SUM	393	00B0AB	XYSLICE	476
0FA007	^FLAGSUM		00C0AB	xSLOPEFIELD	471
0FB007	^SUMVX	393	00D0AB	xPCONTOUR	466
0FC007	^FLAGSUMVX		00E0AB	xDIFFEQ	457
0FD007	^RATSUM	393	00F0AB	xVERSION	475
0FE007	^FTAYL	393	0110AB	xRECT	469
0FF007	^CSTFRACTION?	393	0120AB	xCYLIN	456
100007	^NONRATSUM	393	0130AB	xSPHERE	471
101007	^LINEARAPPLY	425	0150AB	xLININ	463
102007	^linearapply		0160AB	xLIBEVAL	463
103007	^meta_cst?	393	0170AB	xFLASHEVAL	459
104007	^HYPERGEO	393	0180AB	xCONLIB	456
105007	^fk+1/fk		0190AB	xCONST	456
106007	^A/B2PQR	425	01A0AB	xFFT	459
107007	^GOSPER?	425	01B0AB	xIFFT	462
108007	^ZEILBERGER	393	01C0AB	xNDIST	465
109007	^SYMPSI	393	01D0AB	xPSDEV	467
10A007	^sympsi		01F0AB	xPCOV	466
10B007	^SYMPSIN	394	0200AB	xRKF	469
10C007	^sympsin		0210AB	xRKFSTEP	469
10D007	^IBERNOULLI	394	0220AB	xRKFERR	469
10E007	^FLAGRESULTANT	380	0230AB	xRRK	470
10F007	^RESULTANT	379	0240AB	xRRKSTEP	470
110007	^RESULTANTLP	379	0250AB	xRSBERR	470
111007	^RESPSHIFTQ	379	0260AB	xCOND	456
112007	^ADDONEVAR	379	0270AB	XTRACE	474
113007	^IROOTS	382	0280AB	xSRAD	471
114007	^TYPEGAUSSINT?	200,333	0290AB	xSNRM	471
115007	^DTYPEGAUSSINT?	200,333	02A0AB	xRANK	468
116007	^DUPTYPEGAUSSINT?	201,333	02B0AB	xLSQ	464
117007	^PPZZ	424	02C0AB	xEGV	458
118007	^DISTRIB*	368	02D0AB	xEGVL	458
119007	^NONALGERR	404	02E0AB	xSVD	473
11A007	^ALGCASCOMPEVAL	353	02F0AB	xSVL	473
11C007	^%%PSI	394	0300AB	xLU	464
1DC007	^PUSHFLAGS	405	0310AB	xQR	468
1DD007	^POPFLAGS	405	0320AB	xLQ	464
0000AB	xXVOL	476	0330AB	xSCHUR	470
0010AB	xYVOL	476	0340AB	xRREF	470
0020AB	xZVOL	477	0350AB	xRANM	468
0030AB	xXXRNG	476	0360AB	$x \rightarrow ROW$	470
0040AB	xYYRNG	477	0370AB	$\texttt{xROW} {\rightarrow}$	470
0050AB	xEYEPT	459	0380AB	$x{ ightarrow} COL$	455
0060AB	xNUMX	465	0390AB	$\texttt{xCOL} \rightarrow$	455
0070AB	XNUMY	465	03A0AB	$x{ ightarrow} { m DIAG}$	457

0F6007 - 0630B3 611

Addr.	Name	Page	Addr.	Name	Page
03B0AB	$\texttt{xDIAG}{\rightarrow}$	457	0700AB	xXGET	475
03C0AB	xROW-	470	0710AB	xXPUT	476
03D0AB	xROW+	470	0720AB	xMSOLVR	465
03E0AB	xCOL-	455	0730AB	xMINIT	465
03F0AB	xCOL+	455	0740AB	xMITM	465
0400AB	xRSWP	470	0750AB	xMUSER	465
0440AB	xPROOT	467	0760AB	xMCALC	464
0450AB	xPCOEF	466	0770AB	xMROOT	465
0460AB	xPEVAL	466	0050B0	~IFMenuRow1	257
0470AB	xTVM	474	0060B0	~IFMenuRow2	257
0480AB	xTVMBEG	474	0860B0	~grobAlertIcon	90
0490AB	XTVMEND	474	0870B0	~grobCheckKey	90
04A0AB	xTVMROOT	474	0C80B0	~gFldVal	
04B0AB	xAMORT	453	0D80B0	~sFldVal	
04C0AB	xINFORM	462	0DE0B0	$\tilde{n}$ NullBind	117
04D0AB	xCHOOSE	455	0000B1	~DoMsgBox	
04E0AB	xMSGBOX	465	0040B1	~MsgBoxMenu	283
04F0AB	xXSEND	476	0000B3	~Choose	244
0500AB	xXRECV	476	0050B3	~ChooseMenu0	244
0520AB	xTAIL	473	0060B3	~ChooseMenu1	244
0530AB	xSEQ	471	0070B3	~ChooseMenu2	244
0540AB	xDOSUBS	458	0150B3	~BBMoveTo	245
0550AB	$\mathtt{x}\Delta\mathtt{LIST}$	460	0190B3	~BBRecalOff&Disp	245
0560AB	xNSUB	465	0220B3	~BBRunEntryProc	245
0570AB	xENDSUB	458	0230B3	~BBReReadPageSize	245
0580AB	xSTREAM	472	0240B3	~BBReReadHeight	245
0590AB	$ extbf{x}\Sigma  extbf{LIST}$	461	0250B3	~BBReReadCoords	245
05A0AB	$x\Pi$ LIST	466	0260B3	~BBReReadWidth	245
05B0AB	xDOLIST	457	0280B3	~BBRunENTERAction	245
05D0AB	xREVLIST	469	0290B3	~BBRunCanclAction	245
05E0AB	xSORT	471	02F0B3	~BBReDrawBackgr	245
05F0AB	xZFACTOR	477	0370B3	~BBGetNGrob	245
0600AB	xFANNING	459	0380B3	~BBGetNStr	245
0610AB	xDARCY	456	03B0B3	~BBRereadChkEnbl	246
0620AB	xF0 $\lambda$	459	03C0B3	~BBRereadFullScr	246
0630AB	xSIDENS	471	03D0B3	~BReReadMenus	246
0640AB	xTDELTA	473	03E0B3	~BBReReadNElems	246
0650AB	xTINC	473	03F0B3	~BBGetN	246
0660AB	xgmol		04B0B3	~BBIsChecked?	246
0670AB	xlbmol		0520B3	~BBUpArrow	246
0680AB	xrpm		0530B3	~BBDownArrow	246
0690AB	xdB	456	0540B3	~BBSpace	246
06A0AB	xPINIT	466	0590B3	~BBPgDown	246
06B0AB	xDRAW3DMATRIX	458	05A0B3	~BBPgUp	246
06C0AB	$x{ ightarrow}KEYTIME$	463	05B0B3	~BBEmpty?	246
06D0AB	$x$ KEYTIME $\rightarrow$	463	05C0B3	$^{\sim}$ BBGetDefltHeight	246
06E0AB	xXSERV	476	05F0B3	~\$>grobOrGROB	96
06F0AB	xROMUPLOAD	469	0630B3	~ChooseSimple	244
			•		

Addr.	Name	Page	Addr.	Name	Page
0000DD	$x{ ightarrow} LANGUAGE$	463	0190DE	xDISTRIB	
0010DD	$\texttt{xLANGUAGE} {\rightarrow}$	463	01A0DE	xEXP2POW	
0020DD	$x{ ightarrow} FONT$	459	01B0DE	XPOWEXPAND	
0030DD	$\texttt{xFONT} {\longrightarrow}$	459	01C0DE	xTAN2CS2	473
0040DD	$x{ ightarrow}{ m HEADER}$	461	01D0DE	xCIRC	455
0050DD	$x$ HEADER $\rightarrow$	461	01E0DE	xC2P	455
0060DD	$x \rightarrow NDISP$	465	01F0DE	xP2C	
0070DD	xEDIT	458	0200DE	xMSLV	
0080DD	xVISIT	475	0210DE	xDOMAIN	
0090DD	xEDITB	458	0220DE	xSIMPLIFY	471
00A0DD	xVISITB	475	0230DE	xDROITE	
00B0DD	xEQW	458	0240DE	xSTORE	
00C0DD	xFILER	459	0250DE	xDEF	
00D0DD	xFONT8	459	0260DE	xASSUME	
00E0DD	xFONT7	459	0270DE	XUNASSUME	
00F0DD	xFONT6	459	0280DE	xREWRITE	
0100DD	xSREPL	471	0290DE	xINTEGER	
0110DD	$x \rightarrow MINIFONT$	464	02A0DE	xCONSTANTS	
0120DD	$\texttt{xMINIFONT}{\rightarrow}$	465	02B0DE	xHYPERBOLIC	
0130DD	xRENAME	469	02C0DE	xMODULAR	
0140DD	$\mathtt{xUFL1} { o} \mathtt{MINIF}$	474	02D0DE	xPOLYNOMIAL	
0150DD	xDBUG	456	02E0DE	XTESTS	
0160DD	xDISPXY	457	02F0DE	xMATHS	
0000DE	xADDTOREAL	453	0300DE	xCOLLECT	455
0010DE	xSIGMAVX	471	0310DE	xUNASSIGN	
0020DE	xSIGMA	471	0320DE	xHELP	
0030DE	xPsi	467	0330DE	xCASCMD	455
0040DE	xPSI	467	0340DE	xPUSH	
0050DE	xRESULTANT	469	0350DE	xPOP	
0060DE	xIBERNOULLI	461	0360DE	xDEGREE	
0070DE	xGAMMA	460	0370DE	xDEDICACE	
0080DE	xqr		0380DE	XPOTENTIAL	
0090DE	xGRAMSCHMIDT		0390DE	xVPOTENTIAL	
00A0DE	xSYST2MAT		03F0DE	xRCLVX	468
00B0DE	xCHOLESKY		0400DE	xSTOVX	472
00C0DE	xDIAGMAP		0030E0	~BRStoC1	
00D0DE	xISOM		0100E0	~BRbrowse	
00E0DE	xMKISOM		0130E0	~BRoutput	
00F0DE	xKER		0180E0	~BRRclCurRow	
0100DE	xIMAGE		0190E0	~BRRclC1	246
0110DE	xBASIS		0120E4	~MESRclEqn	
0120DE	xIBASIS		0110E7	~UTVUNS1Arg	
0130DE	xAUGMENT		02E0E7	~PCunpack	
0140DE	xPMINI		02F0E7	~UTTYPEEXT0?	
0150DE	xCYCLOTOMIC		01E0E8	~INTEMPOB?	171
0160DE	xSTURM		000100	$x{ ightarrow} H$	478
0170DE	xSTURMAB		001100	$x H \!\! \to \!\!$	478
0180DE	xFDISTRIB		002100	$x{ ightarrow} A$	478

Addr.	Name	Page	Addr.	Name	Page
003100	$xA \rightarrow$	478	00D314	~xRISCH	469
004100	$xA \rightarrow H$	478	00E314	~xDERIV	457
005100	$xH \rightarrow A$	478	00F314	~xDESOLVE	457
006100	$x \rightarrow CD$	478	010314	~xLAP	463
007100	$xCD \rightarrow$	478	011314	~xILAP	462
008100	$xS \rightarrow H$	478	012314	~xLDEC	463
009100	$xH \rightarrow S$	478	013314	~xTEXPAND	473
00A100	$x{ ightarrow} LST$	478	014314	~xLIN	463
00B100	$x{ ightarrow} ALG$	478	015314	~xTSIMP	474
00C100	$x \rightarrow PRG$	478	016314	$\sim$ xLNCOLLECT	464
00D100	$xCOMP \rightarrow$	478	017314	~xEXPLN	459
00E100	$x \rightarrow RAM$	478	018314	~xSINCOS	471
00F100	xSREV	478	019314	~xTLIN	473
010100	xPOKE	478	01A314	$\sim$ xTCOLLECT	473
011100	xPEEK	478	01B314	~xTRIG	474
012100	xAPEEK	478	01C314	~xTRIGCOS	474
013100	xR~SB	478	01D314	~xTRIGSIN	474
014100	xSB~B	479	01E314	~xTRIGTAN	474
015100	xLR~R	479	01F314	~xTAN2SC	473
016100	xS~N	479	020314	~xHALFTAN	461
017100	xLC~C	479	021314	~xTAN2SC2	473
018100	$xASM \rightarrow$	479	022314	~xATAN2S	454
019100	xBetaTesting	479	023314	~xASIN2T	454
01A100	xCRLIB	479	024314	~xASIN2C	454
01B100	xCRC	479	025314	~xACOS2S	453
01C100	xMAKESTR	479	026314	~xDIV2	457
01D100	xSERIAL	479	027314	~xIDIV2	461
01E100	xASM	479	028314	~xQUOT	468
01F100	xER	479	029314	~xIQUOT	462
020100	x→S2	479	02A314	~xREMAINDER	469
021100	xXLIB~	479	02B314	~xIREMAINDER	462
001102	xGETADR	479	02C314	~xGCD	460
002102	xGETNAME	479	02D314	~xLCM	463
003102	xGETNAMES	479	02E314	~xEGCD	458
004102	xGETNEAR	479	02F314	~xIEGCD	462
000314	~xEXPAND	459	030314	~xABCUV	453
001314	~xFACTOR	459	031314	~xIABCUV	461
002314	~xSUBST	472	032314	~xLGCD	463
003314	~xDERVX	457	033314	~xSIMP2	471
004314	~xINTVX	462	034314	~xPARTFRAC	466
005314	~xLIMIT	463	035314	~xPROPFRAC	467
006314	~xTAYLOR0	473	036314	~xPTAYL	467
007314	~xSERIES	471	037314	~xHORNER	461
008314	~xSOLVEVX	471	038314	~xEULER	458
009314	~xPLOT		039314	~xPA2B2	466
00A314	~xPLOTADD	466	03A314	~xCHINREM	455
00B314	~xIBP	461	03B314	~xICHINREM	461
00C314	~xPREVAL	467	03C314	~xISPRIME?	462
		•		<del></del>	<b>-</b>

Addr.	Name	Page	Addr.	Name	Page
03D314	~xNEXTPRIME	465	066314	~xMAP	464
03E314	~xPREVPRIME	467	067314	~xXNUM	475
03F314	~xSOLVE	471	068314	~xXQ	476
040314	~xZEROS	477	069314	~xREORDER	469
041314	~xFCOEF	459	06A314	~xLVAR	464
042314	~xFROOTS	460	06B314	~xFXND	460
043314	~xFACTORS	459	06C314	~xEXLR	459
044314	~xDIVIS	457	06D314	~xLNAME	464
045314	~xTRAN	474	06E314	~xADDTMOD	453
046314	~xHADAMARD	461	06F314	~xSUBTMOD	473
047314	~xrref	470	070314	~xMULTMOD	465
048314	~xREF	469	071314	~xDIVMOD	457
049314	~xAXM	454	072314	~xDIV2MOD	457
04A314	~xAXL	454	073314	~xPOWMOD	467
04B314	~xQXA	468	074314	~xINVMOD	462
04C314	~xAXQ	454	075314	~xGCDMOD	460
04D314	~xGAUSS	460	076314	~xEXPANDMOD	459
04E314	$^{\sim}$ xSYLVESTER	473	077314	~xFACTORMOD	459
04F314	~xPCAR	466	078314	~xRREFMOD	470
050314	~xJORDAN	463	079314	~xMODSTO	465
051314	~xMAD	464	07A314	~xMENUXY	464
052314	$^{\sim}$ xLINSOLVE	463	07B314	$\sim$ xKEYEVAL	463
053314	~xVANDERMONDE	475	07C314	~xGROBADD	460
054314	$^{\sim}$ xHILBERT	461	07D314	~xSCROLL	470
055314	~xLCXM	463	07E314	~xCASCFG	455
056314	~xDIV	457	07F314	~xMAIN	
057314	~xCURL	456	080314	~xBASE	454
058314	~xLAPL	463	080314	xALGB	454
059314	~xHESS	461	081314	~xCMPLX	455
05A314	~xLEGENDRE	463	082314	~xTRIGO	474
05B314	$\sim$ xTCHEBYCHEFF	473	083314	~xMATR	
05C314	$\sim$ xHERMITE	461	084314	~xDIFF	457
05D314	$^{\sim}$ xLAGRANGE	463	085314	~xARIT	453
05E314	~xFOURIER	459	086314	~xSOLVER	471
05F314	~xSIGNTAB	471	087314	~xEXP&LN	
060314	~xTABVAR	473	088314	~xEPSX0	458
061314	~xTABVAL	473	089314	~x?	
062314	~xDIVPC	457	08A314	~x∞	462
063314	~xTRUNC	474	08B314	~xPROMPTSTO	467
064314	~xSEVAL	471	08C314	~xVER	475
065314	$^{\sim}$ xTEVAL	473	1		